Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Μαθηματικών
Μεταπτυχιακό Πρόγραμμα στη Λογική και Θεωρία Αλγορίθμων
και Υπολογισμού

Ευρετικοί Αλγόριθμοι για το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών

Κωνσταντίνος Μάστακας
ΑΜ: 201102

Διπλωματική Εργασία

Επιβλέπων: **Καθ. Αντώνιος Συμβώνης**

Συμβουλευτική – Εξεταστική Επιτροπή

Άρης Παγουρτζής **Γραμματή Πάντζιου** **Αντώνιος Συμβώνης**

Αθήνα 2014

Η παρούσα Διπλωματική Εργασία
εκπονήθηκε στα πλαίσια των σπουδών
για την απόκτηση του
Μεταπτυχιακού Διπλώματος Ειδίκευσης
στη
Λογική και Θεωρία Αλγορίθμων και Υπολογισμού
που απονέμει το
Τμήμα Μαθηματικών
του
Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 27η Μαρτίου 2014 από Εξεταστική Επιτροπή
αποτελούμενη από τους:

Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1. Συμβώνης Α.	Καθηγητής	.....
2. Πάντζιου Γ.	Καθηγήτρια	.....
3. Παγουρτζής Α.	Επικ. Καθηγητής	.....

Περίληψη

Ένας τουρίστας δεν έχει αρκετό χρόνο για να επισκεφτεί όλα τα αξιοθέατα του προορισμού του. Οπότε έχει να επιλέξει ποια θα επισκεφτεί και να σχεδιάσει τις ημερήσιες διαδρομές του ταξιδιού. Το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών (TTDP) αναζητά τις πιο ευχάριστες διαδρομές για τον τουρίστα. Το TTDP μοντελοποιείται σαν το Πρόβλημα του Προσανατολισμού (OP), που είναι NP-hard, ή επεκτάσεις του. Στο OP δίνεται ένα σύνολο κόμβων συσχετισμένων με κέρδος και ζητείται η διαδρομή που συλλέγει το μέγιστο κέρδος, με μήκος φραγμένο από δοσμένο χρονικό περιθώριο. Το TOPTW επεκτείνει το OP αναζητώντας ένα πλήθος διαδρομών, επιτρέποντας την επίσκεψη στους κόμβους σε συγκεκριμένα χρονικά διαστήματα. Το TDTOPTW επεκτείνει το TOPTW θεωρώντας χρονικά εξαρτημένα κόστη ακμών.

Στην παρούσα εργασία, παρουσιάζονται καινοτόμοι ευρετικοί αλγόριθμοι για το TOPTW και το TDTOPTW. Στόχος των αλγορίθμων είναι η παραγωγή σχεδόν βέλτιστων λύσεων σε γρήγορο χρόνο εκτέλεσης. Για το TOPTW προτείνονται οι νέοι ευρετικοί αλγόριθμοι CSCRatio και CSCRoutes, ενώ για το TDTOPTW προτείνονται νέοι ευρετικοί αλγόριθμοι που επεκτείνουν τον CSCRoutes ώστε να χειρίζεται χρονικά εξαρτημένα κόστη ακμών. Οι προτεινόμενοι αλγόριθμοι συγκρίνονται με τους καλύτερους και αποδοτικότερους έως τώρα αλγορίθμους σε προϋπάρχοντα στιγμιότυπα όπως και σε νέα που αναπτύχθηκαν προκειμένου να προσομοιωθούν ρεαλιστικές τουριστικές τοπολογίες.

Λέξεις Κλειδιά: Ευρετικοί Αλγόριθμοι, Σχεδιασμός Τουριστικών Διαδρομών, Χρονικό Περιθώριο, Χρονικά Παράθυρα, Μεγιστοποίηση Κέρδους.

Abstract

A tourist does not have enough time to visit all attractions of the destination. Hence, she has to decide which to visit and plan the daily routes of the trip. Tourist Trip Design Problem (TTDP) asks for the most pleasant routes for the tourist. TTDP is modelled as the Orienteering Problem (OP), which is NP-hard, or extensions of OP. In OP a set of nodes is given, each associated with profit, and asks for the route with the maximum collected profit and length bounded by a given time budget. TOPTW extends OP asking for multiple routes, allowing the visit at nodes to take place within specified time intervals. TDTOPTW extends TOPTW considering time dependent travel costs.

In this thesis novel heuristic algorithms are presented for TOPTW and TDTOPTW. The algorithms aim at producing near optimal solutions within fast execution time. Two new heuristic algorithms CSCRatio and CSCRoutes are presented for TOPTW, while new heuristic algorithms, that extend CSCRoutes to incorporate time dependent travel costs, are proposed for the TDTOPTW. The proposed algorithms are compared with the most efficient until now algorithms in existing instances and in new, created to simulate realistic tourist topologies.

Keywords: Heuristic Algorithms, Tourist Trip Design, Time Budget, Time Windows, Profit Maximization.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή Αντώνιο Συμβώνη, που είναι ο επιβλέπων της διπλωματικής μου εργασίας, για την καθοδήγηση του καθ' όλη την διάρκεια συγγραφής της εργασίας αυτής, όπως επίσης και για όλες τις γνώσεις που μου έχει μεταλαμπαδεύσει συνολικά στην ακαδημαϊκή μου σταδιοδρομία. Θα ήθελα να ευχαριστήσω ιδιαίτερα τους καθηγητές Γραμματή Πάντζιου, Δαμιανό Γαβαλά και Χαράλαμπο Κωνσταντόπουλο για όλη την δουλειά που έχουμε κάνει μαζί και κυρίως γιατί με βοήθησαν να καταλάβω τι σημαίνει έρευνα και πως πρέπει να δουλεύω για να είμαι παραγωγικός. Επίσης, η συνεργασία με τους Γιάννη Τασούλα και Νίκο Βάθη ήταν χαρά μου. Θέλω ακόμα να ευχαριστήσω τον Μιχάλη Μπέκο, την Ευμορφία Αργυρίου και την Χρυσάνθη Ραυτοπούλου για τις πολλές ευχάριστες εκπαιδευτικές ώρες που περάσαμε μαζί. Επιπλέον, θέλω να ευχαριστήσω τον καθηγητή Άρη Παγουρτζή ως μέλος της τριμελούς μου επιτροπής και για την διδασκαλία και την μετάδοση των γνώσεων του σε πολλά μαθήματα του μεταπτυχιακού προγράμματος. Τέλος, θα ήθελα να ευχαριστήσω όλους τους καθηγητές που μου έκαναν μάθημα στο Μ.Π.Λ.Α, καθώς κάθε ένας με τον δικό του ξεχωριστό τρόπο συνέβαλε στην διαμόρφωση της ακαδημαϊκής μου προσωπικότητας.

Μέρος αυτής της δουλειάς έγινε στο πλαίσιο του ευρωπαικού προγράμματος eCOMPASS - EU FP7/2007-2013 (DG CONNECT.H5-Smart Cities and Sustainability), under grant agreement no. 288094.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών (TTDP)

Ένας τουρίστας στις μέρες μας μπορεί να επισκεφτεί εύκολα μακρινούς προορισμούς με πληθώρα αξιοθεάτων. Συνήθως, ο τουρίστας γνωρίζει κάποια από τα σημεία ενδιαφέροντος (POIs) που ταιριάζουν στις προτιμήσεις του, ενώ για τα υπόλοιπα χρειάζεται να ψάξει σε διάφορες ιστοσελίδες και τουριστικά περιοδικά. Οι πληροφορίες που παρέχονται από αυτά είναι γενικού περιεχομένου, οπότε ο ίδιος καλείται να διαλέξει ποια από αυτά ταιριάζουν στα δικά του ενδιαφέροντα. Επίσης, η διάρκεια της παραμονής ενός τουρίστα στον προορισμό του τις περισσότερες φορές δεν επαρκεί για να επισκεφτεί όλα τα αξιοθέατα που θα ήθελε. Οπότε, έχει να επιλέξει ποια θα επισκεφτεί, όπως επίσης και τις ημερήσιες διαδρομές που θα ακολουθήσει, ώστε να μην ξεπερνά το προβλεπόμενο χρονικό περιθώριο κάθε μέρας. Αυτή η διαδικασία περιλαμβάνει τον υπολογισμό των χρονικών κοστών μετάβασης μεταξύ των POIs, όπως επίσης και την κατάλληλη ταξινόμηση των επισκεπτόμενων POIs ώστε να ελαχιστοποιείται το συνολικό χρονικό κόστος κάθε διαδρομής. Όλη αυτή η διαδικασία είναι πολύ απαιτητική για τους τουρίστες.

Για την διευκόλυνση των τουριστών χρησιμοποιούνται Προσωπικοί Ηλεκτρονικοί Τουριστικοί Οδηγοί (PETs) [20, 29, 30, 46, 47]. Κάποιοι PETs προτείνουν ένα σύνολο αξιοθεάτων στους τουρίστες συναφή με τα προσωπικά τους ενδιαφέροντα. Επιπλέον, PETs μπορούν να χρησιμοποιηθούν για τον σχεδιασμό τουριστικών διαδρομών.

Το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών (TTDP) [82] στοχεύει στην εύρεση των πιο ευχάριστων διαδρομών για τον τουρίστα, με βάση τα δικά του ενδιαφέροντα. Οι παραγόμενες διαδρομές πρέπει να ικανοποιούν κάποιους περιορισμούς που θέτει ο ίδιος, π.χ. ένα ημερήσιο χρονικό περιθώριο για επίσκεψη σε POIs. Επιπλέον, οι διαδρομές πρέπει να σέβονται τους περιορισμούς του τουριστικού προορισμού, π.χ. τα ωράρια λειτουργίας των αξιοθεάτων.

Αρκετές εφαρμογές διαδικτύου και κινητών τηλεφώνων έχουν προταθεί για τον σχεδιασμό τουριστικών διαδρομών [21, 32, 63, 81]. Οι εφαρμογές δέχονται ως είσοδο τις προτιμήσεις των χρηστών, λόγου χάρη πόσο ενδιαφέρονται να επισκεφτούν μουσεία, πάρκα και εκκλησίες, όπως επίσης και τους περιορισμούς τους, για παράδειγμα τις ημέρες που θα γίνει η επίσκεψη και τον ημερήσιο διαθέσιμο χρόνο για περιήγηση. Τότε, με βάση τις πληροφορίες του προορισμού που έχουν στην βάση δεδομένων τους, λόγου χάρη το ωράριο λειτουργίας των αξιοθεάτων και τον πίνακα των κοστών μετάβασης μεταξύ των POIs, παράγουν διαδρομές πολύ ευχάριστες για τον χρήστη που σέβονται τους περιορισμούς που έχει θέσει. Ο γρήγορος χρόνος εκτέλεσης είναι πολύ

σημαντικός για τις εφαρμογές καθώς ο χρήστης δεν θα περιμένει παραπάνω από κάποια δευτερόλεπτα για την παραγωγή των ημερήσιων διαδρομών του.

Για την επίλυση του TTDP χρησιμοποιούνται διάφορες μοντελοποιήσεις, ανάλογα με την ακρίβεια που πρέπει να προσεγγιστεί το πρόβλημα. Το TTDP μοντελοποιείται συνήθως σαν το Πρόβλημα Προσανατολισμού (OP) ή μια επέκτασή του. Στο OP δίνεται ένα σύνολο κόμβων συσχετισμένων με κέρδος και το ζητούμενο είναι να παραχθεί η διαδρομή με το μεγαλύτερο κέρδος, έχοντας μήκος φραγμένο από ένα συγκεκριμένο χρονικό περιθώριο. Το TOPTW επεκτείνει το OP αναζητώντας ένα πλήθος διαδρομών, επιτρέποντας όμως την επίσκεψη στους κόμβους μόνο σε συγκεκριμένα χρονικά παράθυρα, που προσομοιώνουν τις ώρες λειτουργίας των POIs. Ακόμα, το TDTOPTW επεκτείνει το TOPTW θεωρώντας χρονικά εξαρτημένα κόστη ακμών. Το OP είναι NP-hard, οπότε μόνο ευρετικές αλγοριθμικές προσεγγίσεις μπορούν να χρησιμοποιηθούν σε εφαρμογές για την επίλυση στιγμιοτύπων του ή των επεκτάσεων του.

1.2 Συνεισφορά της Εργασίας

Στην παρούσα εργασία προτείνονται νέες αλγοριθμικές προσεγγίσεις για το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών. Αρχικά, ερευνώνται τα προβλήματα που μοντελοποιούν το πρόβλημα και οι αλγοριθμικές τεχνικές που έχουν προταθεί για την επίλυσή τους. Ύστερα, εξετάζονται οι δύο μοντελοποιήσεις του TTDP που προσεγγίζουν το πρόβλημα σε όσο το δυνατόν μεγαλύτερο βαθμό. Για κάθε μία από αυτές εισάγονται νέες αλγοριθμικές τεχνικές που παράγουν διαδρομές σχεδόν βέλτιστες για τους τουρίστες. Επιπλέον, οι τεχνικές παράγουν λύσεις με λίγες μετακινήσεις με μέσα μαζικής μεταφοράς. Τέλος, ο χρόνος εκτέλεσης τους είναι αρκετά γρήγορος ώστε να μπορούν να χρησιμοποιηθούν σε εφαρμογές διαδικτύου και κινητών τηλεφώνων.

Αναλυτικά η συνεισφορά της εργασίας έχει ως εξής:

Στο Κεφάλαιο 2 γίνεται ανασκόπηση της βιβλιογραφίας σχετικά με το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών. Μελετώνται τα προβλήματα που μοντελοποιούν συνήθως το πρόβλημα και αναφέρονται οι αλγοριθμικές προσεγγίσεις που έχουν προταθεί για την επίλυση των στιγμιοτύπων τους. Έμφαση δίνεται ιδιαίτερα στις ευρετικές τεχνικές που χρησιμοποιούνται. Το Κεφάλαιο αποτελεί προϊόν συνεργασίας με τους Δαμιανό Γαβαλά, Χαράλαμπο Κωνσταντόπουλο και Γραμματή Πάντζιου. Μέρος του Κεφαλαίου εμφανίζεται στο [33].

Στο Κεφάλαιο 3 προτείνονται δυο νέοι ευρετικοί αλγόριθμοι για το TOPTW, ο CSCRatio και ο CSCRoutes. Οι αλγόριθμοι συγκρίνονται με τον ILS αλγόριθμο των Vansteenwegen et al. [80], που αποτελεί τον πιο αποδοτικό αλγόριθμο επίλυσης του TOPTW σε εφαρμογές. Η σύγκριση γίνεται σε προϋπάρχοντα στιγμιότυπα όπως και σε καινούργια, δημιουργημένα ώστε να προσομοιώνουν ρεαλιστικές τουριστικές τοπολογίες. Το Κεφάλαιο αποτελεί προϊόν συνεργασίας με τους Δαμιανό Γαβαλά, Χαράλαμπο Κωνσταντόπουλο, Γραμματή Πάντζιου και Γιάννη Τασούλα. Μέρος των αποτελεσμάτων έχει δημοσιευτεί στο [34].

Στο Κεφάλαιο 4 εισάγονται τρεις νέοι ευρετικοί αλγόριθμοι για το TDTOPTW, οι TDCSCRoutes, SlackCSCRoutes και AvgCSCRoutes. Οι αλγόριθμοι βασίζονται στον αλγόριθμο CSCRoutes για το TOPTW και συγκρίνονται με την μοναδική προϋπάρχουσα αλγοριθμική προσέγγιση [31]. Η σύγκριση πραγματοποιείται σε καινούργια στιγμιότυπα που δημιουργήθηκαν με βάση την πόλη της Αθήνας. Το Κεφάλαιο αποτελεί προϊόν συνεργασίας με τους Δαμιανό Γαβαλά, Χαράλαμπο Κωνσταντόπουλο, Γραμματή Πάντζιου και Νίκο Βάθη. Μέρος των αποτελεσμάτων έχει δημοσιευτεί στο [35].

Τέλος, στο Κεφάλαιο 5 παρατίθενται τα συμπεράσματα που προέκυψαν από την παρούσα εργασία.

Κεφάλαιο 2

Ανασκόπηση της βιβλιογραφίας

Στο κεφάλαιο αυτό γίνεται ανασκόπηση των προβλημάτων που μοντελοποιούν το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών (TTDP) [82]. Για κάθε ένα από τα προβλήματα, αναφέρονται οι βασικές αλγοριθμικές προσεγγίσεις που χρησιμοποιούνται για την επίλυσή του δίνοντας έμφαση κυρίως στις μεταευρετικές τεχνικές.

Στην πιο απλή εκδοχή του, το TTDP μοντελοποιείται σαν το Πρόβλημα του Προσανατολισμού (OP) [77]. Στο OP δίνεται ένα σύνολο κόμβων συσχετισμένων με κέρδος και ζητείται να παραχθεί η διαδρομή που συλλέγει το μεγαλύτερο δυνατό κέρδος, ενώ ταυτόχρονα το μήκος της δεν ξεπερνάει ένα συγκεκριμένο χρονικό περιθώριο. Το Πρόβλημα του Ομαδικού Προσανατολισμού (TOP) [15] επεκτείνει το OP αναζητώντας ένα πλήθος διαδρομών, ενώ στο TOP με χρονικά παράθυρα (TOPTW) [78] η επίσκεψη σε κάθε κόμβο μπορεί να πραγματοποιηθεί μόνο κατά την διάρκεια ενός χρονικού παραθύρου. Τέλος, στο TDTOPTW [31] θεωρούνται χρονικά εξαρτημένα κόστη ακμών.

Το OP είναι NP-hard [39], οπότε ακριβείς αλγόριθμοι που παράγουν την βέλτιστη λύση, για κάθε στιγμιότυπο του OP ή των επεκτάσεων του, έχουν στην χειρότερη περίπτωση εκθετικό χρόνο εκτέλεσης. Οι ακριβείς αλγόριθμοι για τα προβλήματα που θα αναφερθούν αποφεύγουν την εξαντλητική αναζήτηση χρησιμοποιώντας τεχνικές όπως branch-and-bound [54], branch-and-cut [61] και branch-and-price [6], παραμένοντας όμως εκθετικού χρόνου και άρα μη εφαρμόσιμοι στην πράξη. Για τον λόγο αυτό, χρησιμοποιούνται ευρετικοί αλγόριθμοι πολυωνυμικού χρόνου που δεν παράγουν την βέλτιστη λύση σε κάθε στιγμιότυπο αλλά μια λύση με κέρδος πολύ κοντά στο βέλτιστο. Αρκετοί προσεγγιστικοί αλγόριθμοι [84, 86] έχουν προταθεί, κυρίως για προβλήματα που αναζητούν μια μοναδική διαδρομή. Οι προσεγγιστικοί αλγόριθμοι εγγυώνται ότι για κάθε στιγμιότυπο η παραγόμενη λύση θα απέχει μια καθορισμένη απόσταση από την βέλτιστη δυνατή. Παρ' όλα αυτά, η χρονική πολυπλοκότητα τους είναι συνήθως απαγορευτική για την χρήση τους σε εφαρμογές διαδικτύου (πολυωνυμική μεγάλου βαθμού). Επιπλέον, η αποδοσή τους συνήθως υπολείπεται σε πραγματικά στιγμιότυπα ευρετικών αλγορίθμων που δεν εγγυώνται την παραγωγή λύσεων σε κάποια συγκεκριμένη απόσταση από την βέλτιστη.

Τα προβλήματα που ζητούν την παραγωγή πολλαπλών τουριστικών διαδρομών και απαιτούν γρήγορο χρόνο εκτέλεσης, αντιμετωπίζονται κυρίως με μεταευρετικές τεχνικές. Οι Blum και Roli [9] κατηγοριοποιούν τις μεταευρετικές τεχνικές σε μεθόδους τροχιάς και μεθόδους πληθυσμού. Στις μεθόδους τροχιάς θεωρείται μία μοναδική λύση η οποία κατά την διάρκεια της διαδικασίας εξερευνά τον χώρο των λύσεων με έναν κατάλληλο τρόπο. Συνήθως οι μέθοδοι αυτής της κατηγορίας επεκτείνουν την

τεχνική της Τοπική Αναζήτησης. Κάποιες από τις πιο συνηθισμένες τεχνικές αποτελούν η Simulated Annealing, η Αναζήτηση Tabu, η Αναζήτηση Μεταβλητής Γειτονιάς, η Κατευθυνόμενη Τοπική Αναζήτηση, η Διαδικασία Άπληστης Στοχαστικής Προσαρμόσιμης Αναζήτησης (GRASP) και η Επαναλαμβανόμενη Τοπική Αναζήτηση [87]. Στις μεθόδους πληθυσμού θεωρείται ένα σύνολο λύσεων, το οποίο μεταβάλλεται κατά την διάρκεια της διαδικασίας. Συνηθισμένες τεχνικές αυτής της κατηγορίας αποτελούν οι Εξελικτικοί Αλγόριθμοι και τα συστήματα Ant Colony. Μια εκτενής περιγραφή των μεταευρετικών τεχνικών για προβλήματα συνδυαστικής βελτιστοποίησης μπορεί να βρεθεί στο [9].

Το Κεφάλαιο αυτό δομείται ως εξής: Στην Ενότητα 2.1 παρουσιάζεται το Πρόβλημα του Προσανατολισμού (OP). Το Πρόβλημα του Ομαδικού Προσανατολισμού (TOP) μελετάται στην Ενότητα 2.2. Ενώ, οι αλγοριθμικές προσεγγίσεις για το TOP με χρονικά παράθυρα (TOPTW) παρουσιάζονται στην Ενότητα 2.3. Τέλος, στην Ενότητα 2.4 αναφέρονται οι τεχνικές επίλυσης των στιγμιοτύπων του TOPTW με χρονικά εξαρτημένα κόστη ακμών.

2.1 Το Πρόβλημα του Προσανατολισμού (OP)

Ο Τσιλιγκιρίδης [77] όρισε το OP περιγράφοντας ένα παιχνίδι στην ύπαιθρο. Στο παιχνίδι, υπάρχουν τοποθεσίες με πόντους και ένα καθορισμένο χρονικό περιθώριο. Ξεκινώντας από την αφετηρία, οι παίκτες πρέπει να φτάσουν στον τερματισμό μέσα στο χρονικό περιθώριο έχοντας μαζέψει όσο περισσότερους πόντους μπορούν. Το OP συναντάται στην βιβλιογραφία επίσης ως το Πρόβλημα του Επιλεκτικού Περιοδεύοντος Πωλητή (Selective Traveling Salesman Problem) [53] και σαν το Πρόβλημα της Μέγιστης Συλλογής (Maximum Collection Problem) [43].

Ένα στιγμιότυπο του OP αποτελείται από (i) έναν (κατευθυνόμενο ή μη) γράφο $G = (V, E)$, όπου $V = \{u_1, u_2, \ldots, u_N\}$ είναι το σύνολο των κορυφών και E το σύνολο των ακμών, τέτοιος ώστε κάθε κορυφή u_i σχετίζεται με ένα μη αρνητικό κέρδος p_i και κάθε ακμή e έχει ένα μη αρνητικό κόστος διαπέρασης c_e, (ii) έναν αρχικό κόμβο $s = u_1$ και έναν τελικό κόμβο $t = u_N$ και (iii) ένα χρονικό περιθώριο B. Ο στόχος του προβλήματος είναι να βρεθεί η $s - t$ διαδρομή με συνολικό μήκος το πολύ B που να συλλέγει το μέγιστο κέρδος από τις κορυφές που επισκέπτεται. Η διαδρομή μπορεί να επισκεφτεί έναν κόμβο αρκετές φορές, αλλά το κέρδος κάθε επισκεπτόμενου κόμβου μετράται μόνο μία φορά στο συνολικό συλλεγμένο κέρδος.

Διαφορετικές εκδοχές του OP έχουν μελετηθεί στην βιβλιογραφία. Το OP διαφοροποιείται ανάλογα με το αν ο γράφος είναι μη κατευθυνόμενος [5, 77] ή κατευθυνόμενος (directed OP) [65]. Επιπλέον, έχουν εξεταστεί προβλήματα όπου δίνεται μόνο ένας αρχικός και όχι τελικός κόμβος, δηλαδή η διαδρομή μπορεί να τελειώνει σε οποιονδήποτε κόμβο του γράφου, (rooted OP) [3, 19]. Ακόμα, μπορεί να μην υπάρχει ούτε καθορισμένος αρχικός κόμβος, δίνοντας την ελευθερία στην διαδρομή να ξεκινήσει και να τερματίσει σε οποιονδήποτε κόμβο του γράφου, (unrooted OP) [37].

Οι Golden et al. [39] έδειξαν ότι το OP είναι NP-hard ανάγοντας την αποφαντική εκδοχή του Προβλήματος του Περιοδεύοντος Πωλητή (DTSP), που είναι NP-complete [67] στο OP. Το DTSP ρωτάει αν δοθέντος ενός γράφου G με κόστη στις ακμές και ενός στόχου C υπάρχει κύκλος που να περνά από όλες τις κορυφές του γράφου με μήκος το πολύ C. Το DTSP ανάγεται στο OP ως εξής: δεδομένου του στιγμιοτύπου του DTSP κατασκευάζεται ένα στιγμιότυπο του OP με τον ίδιο γράφο και κόστη ακμών, όπου κάθε κόμβος έχει κέρδος 1 και οι s, t ταυτίζονται σε έναν τυχαίο κόμβο του γράφου, ενώ το χρονικό περιθώριο ισούται με C. Τότε, η απάντηση στο στιγμιότυπο του DTSP είναι ναι αν και μόνο αν το βέλτιστο μονοπάτι του OP έχει κέρδος ίσο με το πλήθος

των κορυφών του G.

Ακριβείς αλγόριθμοι για το OP που εφαρμόζουν τεχνικές branch-and-bound δίνονται στα [53, 70], ενώ τεχνικές branch-and-cut εφαρμόζονται στα [26, 36].

Αρκετοί προσεγγιστικοί αλγόριθμοι προτείνονται για την επίλυση του OP. Στους προσεγγιστικούς αλγορίθμους για το OP τα κέρδη στους κόμβους θεωρούνται ίσα με 1. Αυτή η θεώρηση μπορεί να χρησιμοποιηθεί καθώς όπως αποδείχτηκε στα [48, 16] ένας $a-$προσεγγιστικός αλγόριθμος για το OP με μοναδιαία κέρδη στους κόμβους παράγει έναν $a(1+o(1))-$προσεγγιστικό αλγόριθμο για το OP με αυθαίρετα θετικά κέρδη στους κόμβους. Επίσης, ένα σημαντικό αποτέλεσμα για την προσεγγισιμότητα του OP αποτελεί το γεγονός ότι το rooted OP είναι APX-hard και ότι δεν μπορεί να προσεγγιστεί με λόγο προσέγγισης μικρότερο από $\frac{1481}{1480}$ [8]. Για μη κατευθυνόμενους γράφους έχουν προταθεί προσεγγιστικοί αλγόριθμοι σταθερού λόγου, ενώ για κατευθυνόμενους, οι λόγοι προσέγγισης είναι συναρτήσεις του μεγέθους του γράφου (π.χ. [16]).

Οι Arkin et al. [3] έδωσαν έναν $(2+\epsilon)$- προσεγγιστικό αλγόριθμο για το rooted OP θεωρώντας τους κόμβους ως σημεία του επιπέδου. Αργότερα οι Blum et al. [7, 8] εισήγαγαν έναν $4-$προσεγγιστικό αλγόριθμο για το rooted OP σε μη κατευθυνόμενους γράφους. Οι Bansal et al. [5] βελτίωσαν αυτό το αποτέλεσμα παράγοντας έναν 3-προσεγγιστικό αλγόριθμο για το OP σε μετρικούς χώρους. Για γράφους όπου οι κόμβοι είναι σημεία στο $\mathbb{R}^d$ και τα κόστη των ακμών είναι ευκλείδιες αποστάσεις παρουσιάζεται ένα PTAS από τους Chen et al. [19]. Οι Chekuri et al. [16] πρότειναν έναν $(2+\epsilon)-$προσεγγιστικό αλγόριθμο για το OP σε μη κατευθυνόμενο γράφο με χρόνο εκτέλεσης $n^{O(1/\epsilon)}$. Επιπλέον, στο ίδιο άρθρο προτάθηκε ένας $O(\log^2 \mathrm{OPT})$-προσεγγιστικός αλγόριθμος για το OP σε κατευθυνόμενο γράφο, όπου με OPT συμβολίζεται το πλήθος των κόμβων στην βέλτιστη διαδρομή. Τέλος, οι Nagarajan και Ravi [65] πρότειναν έναν $O(\frac{\log^2 n}{\log \log n})$-προσεγγιστικό αλγόριθμο για το OP σε κατευθυνόμενο γράφο.

Για το OP, έχουν επίσης προταθεί αρκετοί ευρετικοί και μεταευρετικοί αλγόριθμοι. Ο Τσιλιγκιρίδης [77] πρότεινε έναν στοχαστικό και έναν αιτιοκρατικό αλγόριθμο. Ο στοχαστικός βασίζεται σε Monte-Carlo τεχνικές κατασκευάζοντας ένα πλήθος διαδρομών και επιλέγοντας εκείνη με το μεγαλύτερο κέρδος, ενώ, στον αιτιοκρατικό ο χώρος διαμερίζεται σε κύκλους που περιορίζουν τις διαδρομές. Οι Golden et al. [39] πρότειναν έναν ευρετικό αλγόριθμο στον οποίο θεωρείται το κέντρο βάρους της διαδρομής. Οι κόμβοι με τον μεγαλύτερο λόγο του κέρδους προς την απόσταση τους από το κέντρο βάρους της διαδρομής εισάγονται, στην θέση με το ελάχιστο κόστος εισαγωγής. Η προηγούμενη προσέγγιση βελτιώνεται στο [38], δίνοντας σε ένα σύνολο κόμβων μια ανταμοιβή και σε ένα άλλο σύνολο μια τιμωρία. Στο [69] προτείνεται ένας ευρετικός αλγόριθμος τεσσάρων φάσεων. Στην πρώτη φάση εισάγονται νέοι κόμβοι στην διαδρομή, στην δεύτερη η σειρά επίσκεψης των κόμβων αλλάζει για να εξοικονομηθεί χρόνος, ενώ στην τρίτη αφαιρείται ένα πλήθος κόμβων από την διαδρομή. Τέλος, στην τέταρτη φάση γίνεται προσπάθεια να εισαχθούν νέοι μη επισκεπτόμενοι κόμβοι στην διαδρομή. Οι Wang et al. [85] προτείνουν έναν αλγόριθμο εφαρμόζοντας ένα συνεχές Hopfield νευρωνικό δίκτυο. Ο ευρετικός αλγόριθμος που προτείνεται στο [14] διαμερίζει τους κόμβους σε διαδρομές που σέβονται το επιτρεπόμενο χρονικό περιθώριο. Επαναληπτικά, κόμβοι ανταλλάσσονται μεταξύ των διαδρομών ώστε να προκύψουν καλύτερες λύσεις, ενώ στο τέλος τους αλγορίθμου εφαρμόζεται ένα βήμα διαταραχής. Τέλος, οι Gendreau et al. [37] πρότειναν έναν μεταευρετικό αλγόριθμο Αναζήτησης Tabu για το unrooted OP. Ο αλγόριθμος επαναληπτικά εισάγει ομάδες από κόμβους ή αφαιρεί ένα σύνολο διαδοχικά επισκεπτόμενων στην διαδρομή.

2.2 Το Πρόβλημα του Ομαδικού Προσανατολισμού (TOP)

Το TOP [15] επεκτείνει το OP, αναζητώντας ένα σύνολο διαδρομών αντί για μία. Πιο συγκεκριμένα, ένα στιγμιότυπο του TOP παίρνει μια επιπλέον παράμετρο από ένα στιγμιότυπο του OP, έναν φυσικό αριθμό K που δηλώνει το πλήθος των διαδρομών που πρέπει να παραχθούν. Το ζητούμενο είναι να βρεθούν οι K διαδρομές που ξεκινούν από την αφετηρία, τερματίζουν στον προορισμό σεβόμενες το διαθέσιμο χρονικό περιθώριο και συλλέγουν το μέγιστο δυνατό κέρδος από τους κόμβους. Το TOP εισήχθηκε για πρώτη φορά ως το Πρόβλημα της Μέγιστης Συλλογής από Πολλαπλές Περιοδείες (Multiple Tour Maximum Collection Problem) από τους Butt και Cavalier [12].

Οι Butt et al. [13] προτείνουν έναν ακριβή αλγόριθμο για το TOP που συνδυάζει τεχνικές branch-and-bound και column generation. Ένας ακόμα ακριβής αλγόριθμος δίνεται από τους Boussier et al. [11] χρησιμοποιώντας τεχνική brach-and-price.

Πολλοί ευρετικοί και μεταευρετικοί αλγόριθμοι έχουν προταθεί για το TOP. Οι Butt και Cavalier [12] προτείνουν έναν ευρετικό αλγόριθμο στον οποίο κάθε ζεύγος κόμβων έχει ένα βάρος εισαγωγής. Σε κάθε βήμα του αλγορίθμου εισάγεται ο κόμβος που ανήκει στο ζεύγος με το μέγιστο βάρος στην διαδρομή που ανήκει το ζευγάρι του. Οι Chao et al. [15] προτείνουν έναν μεταευρετικό αλγόριθμο που επεκτείνει τον αλγόριθμο [14] για το OP, θεωρώντας K μονοπάτια και χρησιμοποιώντας δύο βήματα διαταραχής. Ένας αλγόριθμος Αναζήτησης Tabu προτείνεται στο [76] που περιλαμβάνει τρία βασικά βήματα, τα βήματα της αρχικοποίησης, της βελτίωσης και της αποτίμησης. Μεταευρετικοί αλγόριθμοι προτείνονται επίσης από τους Archetti et al. [2]. Οι συγγραφείς ορίζουν ένα σύνολο από γειτονιές των λύσεων του TOP και προτείνουν αλγορίθμους Αναζήτησης Tabu και Αναζήτησης Μεταβλητής Γειτονιάς. Οι Vansteenwegen et al. [79] προτείνουν έναν αλγόριθμο Κατευθυνόμενης Τοπικής Αναζήτησης. Στην λύση του προβλήματος εφαρμόζεται τοπική αναζήτηση, η οποία ακολουθείται από μία Κατευθυνόμενη Τοπική Αναζήτηση ώστε να παραχθούν λύσεις υψηλότερης ποιότητας. Επιπλέον, στο [83] οι ίδιοι συγγραφείς προσεγγίζουν το πρόβλημα χρησιμοποιώντας Σκιώδη Αναζήτηση Μεταβλητής Γειτονιάς. Οι Souffriau et al. [74] προτείνουν έναν μεταευρετικό αλγόριθμο GRASP. Ο αλγόριθμος επαναληπτικά δημιουργεί μια καινούργια λύση, εισάγοντας κόμβους με ένα άπληστο κριτήριο, η οποία βελτιώνεται ως προς το κέρδος και την χρονική της διάρκεια εφαρμόζοντας τοπική αναζήτηση. Επιπλέον, στο [75] προτείνεται μια αλγοριθμική προσέγγιση που συνδυάζει την GRASP και την μεταευρετική τεχνική της Επανασύνδεσης Μονοπατιών (Path Relinking).

Εκτός από μεταευρετικές τεχνικές τροχιάς, προτείνονται επίσης αρκετοί μεταευρετικοί αλγόριθμοι πληθυσμού για την επίλυση του TOP. Οι Ke et al. [44] προσεγγίζουν το πρόβλημα χρησιμοποιώντας τεχνικές Ant Colony, θεωρώντας ένα πλήθος μυρμηγκιών. Σε κάθε επανάληψη μια εφικτή λύση του προβλήματος δημιουργείται για το καθένα, η οποία κατασκευάζεται με μία από τέσσερις προτεινόμενες μεθόδους, ενώ μετά την κατασκευή των λύσεων ακολουθείται τοπική αναζήτηση και ανανέωση των ιχνών της φερορμόνης που παράγουν τα μυρμήγκια. Ένας Μιμητικός Γενετικός Αλγόριθμος για το TOP δίνεται από τους Bouly et al. [10]. Ένα σύνολο χρωμοσωμάτων θεωρείται, όπου το καθένα παράγει μια εφικτή λύση του προβλήματος. Νέα χρωμοσώματα παράγονται χρησιμοποιώντας τεχνικές διασταύρωσης συνδυασμένες με τεχνικές τοπικής αναζήτησης. Τέλος, οι προσεγγίσεις των Muthuswamy et al. [64] και Dang et al. [23] χρησιμοποιούν τεχνικές particle swarm optimization [45]. Στις συγκεκριμένες προσεγγίσεις θεωρείται ένας πληθυσμός μορίων (particles), από τα οποία προκύπτουν εφικτές λύσεις του προβλήματος. Τότε, χρησιμοποιώντας τεχνικές particle swarm optimization και τοπικής αναζήτησης, προκύπτουν υψηλού επιπέδου λύσεις του προβλήματος.

2.3 Το TOP με χρονικά παράθυρα (TOPTW)

Το TOPTW [78] επεκτείνει το TOP, επιτρέποντας την επίσκεψη στους κόμβους μόνο κατά την διάρκεια των χρονικών τους παραθύρων. Τα χρονικά παράθυρα προσομοιώνουν το ωράριο λειτουργίας των αξιοθεάτων, π.χ. αν ένα μουσείο είναι ανοικτό από τις 09:00 έως τις 14:00, τότε το χρονικό παράθυρο του κόμβου που μοντελοποιεί το μουσείο είναι [09:00-14:00]. Η ειδική περίπτωση που ζητείται μία μοναδική διαδρομή αποτελεί το OP με χρονικά παράθυρα (OPTW).

Για το OPTW προτάθηκαν δύο ακριβείς αλγόριθμοι δυναμικού προγραμματισμού από τους Righini και Salani [72]. Ένας $(3\log^2 n)$-προσεγγιστικός αλγόριθμος για το OPTW προτάθηκε από τους Bansal et al. [5], ενώ οι Chekuri et al. [16] έδειξαν ότι ένας α-προσεγγιστικός αλγόριθμος για το OP παράγει έναν $O(\alpha \max\{\log \mathrm{OPT}, \log L\})$-προσεγγιστικό αλγόριθμο για το OPTW, όπου OPT είναι το πλήθος των επισκεπτόμενων κόμβων στο βέλτιστο μονοπάτι και L ο λόγος του μεγίστου προς το ελάχιστο χρονικό παράθυρο. Επιπλέον, προσεγγιστικοί αλγόριθμοι για διάφορες παραλλαγές του προβλήματος προτείνονται από τους Chekuri και Kumar [17], Chekuri και Pal [18] και Frederickson et al. [28]. Εκτός από τους προσεγγιστικούς αλγορίθμους, δυο ευρετικοί αλγόριθμοι για το OPTW προτείνονται από τους Kantor και Rosenwein [42]. Στον πρώτο εισάγονται διαδοχικά νέοι κόμβοι στην διαδρομή με ένα άπληστο κριτήριο, ενώ στο δεύτερο κάποιες υποδιαδρομές του προβλήματος κατασκευάζονται χρησιμοποιώντας μια κατά βάθος αναζήτηση.

Για το TOPTW, οι Labadi et al. [49] προτείνουν έναν μεταευρετικό αλγόριθμο Αναζήτησης Μεταβλητής Γειτονιάς. Οι Lin et al. [58] προτείνουν έναν μεταευρετικό αλγόριθμο Αναζήτησης Tabu. Τρεις γειτονιές θεωρούνται, ενώ σε κάθε βήμα του αλγορίθμου μια από αυτές χρησιμοποιείται με ίση πιθανότητα για την εύρεση της καλύτερης γειτονικής λύσης. Οι Vansteenwegen et al. [80] προτείνουν έναν αλγόριθμο Επαναλαμβανόμενης Τοπικής Αναζήτησης, ο οποίος κατασκευάζει την λύση εισάγοντας διαδοχικά νέους κόμβους στις διαδρομές με ένα άπληστο κριτήριο. Επιπλέον, η λύση διαταράσσεται αφαιρώντας ένα πλήθος κόμβων από κάθε διαδρομή, ώστε να υπάρξει χώρος για μετέπειτα εισαγωγή καινούργιων κόμβων που θα βελτιώσουν την ποιότητά της. Ένα Σύστημα Ant Colony περιγράφεται από τους Montemanni και Gambardella [62]. Μια λύση κατασκευάζεται για κάθε ένα από τα μυρμήγκια, η οποία βελτιώνεται χρησιμοποιώντας τοπική αναζήτηση. Οι Labadi et al. [50], [51] προτείνουν μια αλγοριθμική προσέγγιση που συνδυάζει την GRASP με την Εξελικτική Τοπική Αναζήτηση. Μια αρχική λύση παράγεται από την GRASP, που βελτιώνεται μέσω τοπικής αναζήτησης. Ύστερα, εκτελείται η Εξελικτική Τοπική Αναζήτηση παράγοντας ένα πλήθος λύσεων που προκύπτουν από την προηγούμενη με ένα βήμα διαταραχής και βελτιώνοντας κάθε μία από αυτές με χρήση τοπικής αναζήτησης. Τέλος, οι Hu και Lim [40] προτείνουν έναν μεταευρετικό αλγόριθμο που συνδυάζει τρεις διαδικασίες. Στις δύο πρώτες διαδικασίες εκτελείται Τοπική Αναζήτηση και Simulated Annealing αντίστοιχα, ενώ το σύνολο των διαδρομών που παράγεται από κάθε μία από αυτές εισάγεται σε ένα άλλο αποθηκευτικό σύνολο. Στην τρίτη διαδικασία οι διαδρομές που υπάρχουν στο αποθηκευτικό σύνολο συνδυάζονται κατάλληλα ώστε να προκύψει η εφικτή λύση του προβλήματος με το μέγιστο κέρδος.

2.4 Το χρονικά εξαρτημένο TOPTW (TDTOPTW)

Το TDTOPTW [31] επεκτείνει το TOPTW θεωρώντας χρονικά εξαρτημένα κόστη ακμών. Η εισαγωγή χρονικά εξαρτημένων κοστών μετακίνησης μεταξύ δύο σημείων μοντελοποιεί ρεαλιστικά προβλήματα σε πόλεις. Αυτό συμβαίνει καθώς οι μετακινήσεις

των μέσων μαζικής μεταφοράς είναι συνήθως πιο πυκνές σε ώρες αιχμής, ενώ ταυτόχρονα η κυκλοφοριακή συμφόρηση μπορεί να οδηγήσει σε καθυστερήσεις σε διαφορετικές χρονικές στιγμές της ημέρας.

Η ειδική περίπτωση του TDTOPTW με μία διαδρομή και χωρίς χρονικά παράθυρα (TDOP) μελετήθηκε από τους Fomin και Lingas [27]. Στο συγκεκριμένο άρθρο παρουσιάστηκαν προσεγγιστικοί αλγόριθμοι για το πρόβλημα, που όμως δεν έχουν πολυωνυμικό χρόνο εκτέλεσης. Για το TDOP και το TDTOP (την ειδική περίπτωση του TDTOPTW χωρίς χρονικά παράθυρα) προτείνονται δύο ακριβείς αλγόριθμοι εκθετικού χρόνου στα [56] και [55], αντίστοιχα. Οι Abbaspour et al. [1] προτείνουν έναν ευρετικό γενετικό αλγόριθμο για το TDOPTW (το TDTOPTW που αναζητάει μία διαδρομή), σε αστικές περιοχές με χρήση μέσων μαζικής μεταφοράς. Ο αλγόριθμος χρησιμοποιεί σαν υπορουτίνα του έναν άλλο γενετικό αλγόριθμο για τον υπολογισμό των συντομότερων διαδρομών μεταξύ των αξιοθεάτων της πόλης.

Η μόνη αλγοριθμική δουλειά που μελετάει το TDTOPTW είναι η δουλειά των Garcia et al. [30], [31]. Οι συγγραφείς βασιζόμενοι στον ILS αλγόριθμο [80] για το TOPTW προτείνουν δύο διαφορετικές κατευθύνσεις για την επίλυση του TDTOPTW. Στην πρώτη κατεύθυνση ένα τυχαίο στιγμιότυπο του TDTOPTW ανάγεται σε ένα στιγμιότυπο του TOPTW αντικαθιστώντας τα χρονικά εξαρτημένα κόστη των ακμών από τα αντίστοιχα μέσα κόστη τους. Ύστερα, μια λύση για το παραγμένο στιγμιότυπο του TOPTW δημιουργείται χρησιμοποιώντας τον ILS αλγόριθμο. Τέλος, η λύση αυτή μετατρέπεται σε μια εφικτή λύση για το TDTOPTW, εισάγοντας τα κατάλληλα χρονικά εξαρτημένα κόστη στις ακμές των διαδρομών. Αν αυτό το επιδιορθωτικό βήμα οδηγήσει σε μη έγκυρη λύση, λόγω των χρονικών περιορισμών, τότε οι κόμβοι που δεν ικανοποιούν τα χρονικά τους παράθυρα απομακρύνονται από την λύση. Η δεύτερη προσέγγιση λαμβάνει υπ' όψιν, τους χρονικά εξαρτημένους χρόνους των ακμών, στα βήματα εισαγωγής και διαταραχής του αλγορίθμου, θεωρώντας ότι οι χρονικές στιγμής άφιξης των μέσων μαζικής μεταφοράς στους σταθμούς είναι περιοδικές και ταυτόχρονα οι χρόνοι μετάβασης μεταξύ των σταθμών παραμένουν ίδιοι κατά την διάρκεια της ημέρας. Με βάση αυτήν την παραδοχή η εισαγωγή ενός κόμβου μπορεί να πραγματοποιηθεί χρησιμοποιώντας ένα γρήγορο τοπικό κριτήριο παρόμοιο με του ILS. Για την δεύτερη κατεύθυνση προτείνονται διάφοροι τρόποι μοντελοποίησης των ακμών, ανάλογα με το αν υπάρχει γραμμή μαζικής μεταφοράς που να περνάει από σταθμούς κοντά στους κόμβους του γράφου ή η μετακίνηση μεταξύ των δύο κόμβων προϋποθέτει την χρήση τουλάχιστον δύο συγκοινωνιακών γραμμών.

Κεφάλαιο 3

Ευρετικοί αλγόριθμοι για το TOPTW

Το TOPTW [78] αποτελεί κατάλληλο πρότυπο για το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών. Οι αλγόριθμοι που μπορούν να χρησιμοποιηθούν σε υπηρεσίες διαδικτύου για την παραγωγή τουριστικών διαδρομών πρέπει να έχουν γρήγορο χρόνο εκτέλεσης ($\sim$ 1 δευτερόλεπτο) παράγοντας ταυτόχρονα πολύ ευχάριστες διαδρομές για τον χρήστη με βάση τις δικές του προτιμήσεις και περιορισμούς. Ο πιο κατάλληλος αλγόριθμος έως τώρα είναι ο ILS αλγόριθμος των Vansteenwegen et al. [80]. Ο ILS είναι ένας μεταευρετικός αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης [59] για το TOPTW που παράγει σχεδόν βέλτιστες λύσεις σε λίγο χρόνο εκτέλεσης. Ο ILS βρίσκει ένα σύνολο διαδρομών, εισάγοντας διαδοχικά ένα καινούργιο σημείο ενδιαφέροντος POI σε κάθε διαδρομή. Το 'βέλτιστο' POI για εισαγωγή συνδυάζει μεγάλη ευχαρίστηση για τον τουρίστα και ταυτόχρονα μικρή χρονική επιβάρυνση για την διαδρομή. Το μειονέκτημα της προσέγγισης αυτής είναι ότι ένα POI εισάγεται χωρίς να λαμβάνεται υπ' όψιν αν αυτή η εισαγωγή θα διευκολύνει την εισαγωγή 'καλών' επόμενων POIs ή θα την κάνει πιο δύσκολη. Αυτό έχει ως αποτέλεσμα μια λύση είτε να παγιδεύεται σε απομονωμένα POIs μεγάλης ωφελιμότητας είτε να μην μπορεί να επισκεφτεί απόμακρες γειτονιές με πολλά σημαντικά POIs λόγω έλλειψης χρόνου.

Σε αυτό το κεφάλαιο προτείνονται δύο νέοι ευρετικοί αλγόριθμοι για το TOPTW, ο CSCRatio και ο CSCRoutes. Οι αλγόριθμοι βελτιώνουν τα μειονεκτήματα του ILS χρησιμοποιώντας ομαδοποίηση (clustering) [41] των POIs με βάση την γεωμετρική τους θέση. Ο κύριος στόχος των αλγορίθμων είναι να ευνοήσουν την επίσκεψη απομακρυσμένων περιοχών με πλήθος αξιοθεάτων, βελτιώνοντας έτσι την ποιότητα των παραγόμενων λύσεων. Ταυτόχρονα, ο χρόνος εκτέλεσης πρέπει να διατηρείται ανεκτός από τους χρήστες της εφαρμογής. Επιπλέον, οι αλγόριθμοι έχουν στόχο να μειώσουν τις μετακινήσεις μεταξύ απομακρυσμένων σημείων. Αυτές οι μετακινήσεις είναι συνήθως επιβαρυντικές οικονομικά καθώς απαιτούν χρήση συγκοινωνίας. Αντίθετα οι μετακινήσεις μεταξύ κοντινών τοποθεσιών είναι οικονομικές και οικολογικές καθώς συνήθως γίνονται περπατώντας.

Το κεφάλαιο αυτό δομείται ως εξής: Ο μαθηματικός ορισμός του TOPTW δίνεται στην Ενότητα 3.1. Στην Ενότητα 3.2 παρουσιάζεται η γενική τεχνική της Επαναλαμβανόμενης Τοπικής Αναζήτησης. Ο ILS αλγόριθμος των Vansteenwegen et al. [80] περιγράφεται στην Ενότητα 3.3. Οι αλγορίθμοι CSCRatio και CSCRoutes παρουσιάζονται στην ενότητα 3.4. Τα πειραματικά αποτελέσματα των αλγορίθμων παρουσιάζονται στην Ενότητα 3.5, ενώ τα συμπεράσματα του κεφαλαίου, παρατίθενται στην Ενότητα 3.6.

3.1 Μαθηματικός Ορισμός του TOPTW

Το TOPTW είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης [66] και πιο συγκεκριμένα ένα πρόβλημα δρομολόγησης κορυφών με κέρδη [25] που ορίζεται ως εξής: Τα δεδομένα του είναι:

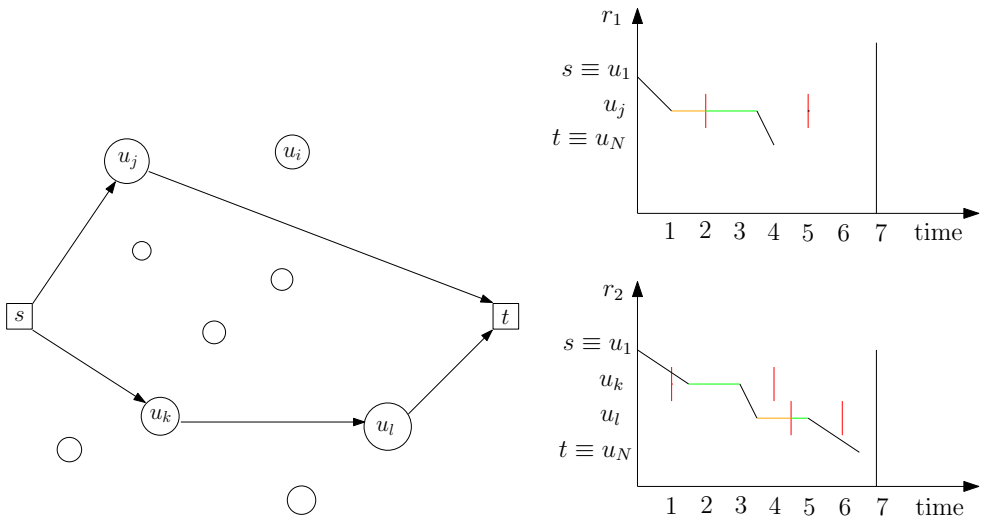
- Ένας θετικός ακέραιος K που δηλώνει το πλήθος των διαδρομών που πρέπει να κατασκευαστούν.
- Ένας πλήρης κατευθυνόμενος γράφος $G = (V, A)$ όπου $V = \{u_1, u_2, \ldots, u_N\}$ είναι το σύνολο των κορυφών του, τέτοιος ώστε:
  - κάθε κορυφή u_i σχετίζεται με ένα κέρδος p_i και ένα χρόνο επίσκεψης v_i τέτοια ώστε η επίσκεψη στον κόμβο u_i συνοδεύεται από συλλογή κέρδους p_i και προϋποθέτει την παραμονή στον κόμβο για χρόνο v_i.
  - κάθε κορυφή u_i έχει ένα χρονικό παράθυρο για κάθε μέρα $[R_{im}, D_{im}], m = 1, 2, \ldots, K$ με $R_{im} \leq D_{im}$ μέσα στο οποίο μπορεί να ξεκινήσει η επίσκεψη σε αυτόν τον κόμβο.
  - Κάθε κορυφή u_i σχετίζεται με μια γεωγραφική τοποθεσία, έχοντας συντεταγμένες $[x_i, y_i]$.
  - κάθε ακμή (u_i, u_j) του γράφου έχει ένα κόστος διαπέρασης c_{ij}.
- Κάθε διαδρομή $r_i, i = 1, 2, \ldots, K$ έχει ένα μέγιστο επιτρεπόμενο μήκος $B_i, i = 1, \ldots, K$.

Μια εφικτή λύση του προβλήματος είναι ένα σύνολο K μονοπατιών $r_1, r_2, \ldots, r_K$ με αφετηρία τον κόμβο $s \equiv u_1$ και τερματισμό τον κόμβο $t \equiv u_N$. Το μήκος κάθε διαδρομής $r_i, i = 1, 2, \ldots, K$ πρέπει να είναι το πολύ B_i. Επιπλέον, κάθε κόμβος διαφορετικός από τους u_1 και u_N ανήκει το πολύ σε μία διαδρομή, ενώ η επίσκεψη σε αυτόν γίνεται μόνο κατά την διάρκεια του χρονικού του παραθύρου. Ο στόχος του προβλήματος είναι να βρεθεί η εφικτή λύση με το μέγιστο δυνατό συλλεγμένο κέρδος από τους κόμβους που επισκέπτεται.

Μια εφικτή λύση του TOPTW εμφανίζεται στην Εικόνα 3.1. Δύο διαδρομές αναζητούνται, που καθεμία έχει χρονικό περιθώριο 7 ώρες. Η πρώτη διαδρομή φτάνει στον κόμβο u_j μετά από 1 ώρα ταξιδιού ($c_{1j} = 1$), ενώ η επίσκεψη στον u_j ξεκινάει μετά από μια ώρα αναμονής (το χρονικό παράθυρο του είναι $[2, 5]$) και έχει διάρκεια 1.5 ώρα ($v_j = 1.5$). Τέλος, το κόστος μετακίνησης από τον u_j στον t είναι μισή ώρα. Αντίστοιχα, για την δεύτερη διαδρομή το χρονικό παράθυρο των u_k και u_l είναι $[1, 4]$ και $[4.5, 6]$ αντίστοιχα. Οι χρόνοι παραμονής στους u_k και u_l είναι 1.5 και 0.5 ώρα αντίστοιχα, ενώ για τα χρονικά κόστη ισχύει $c_{1k} = 1.5, c_{kl} = 0.5, c_{lN} = 0.5$. Με βάση τα παραπάνω, η άφιξη στον κόμβο u_k γίνεται μετά από 1.5 ώρα ταξιδιού και η επίσκεψη στον κόμβο ξεκινάει αμέσως. Ο κόμβος u_j προσεγγίζεται μετά από 3.5 ώρες ταξιδιού, ενώ η επίσκεψη στον κόμβο αυτό αρχίζει μετά από μια ώρα αναμονής. Τέλος, μετά από 6.5 ώρες ταξιδιού η διαδρομή καταλήγει στον τελικό προορισμό.

3.2 Η τεχνική της Επαναλαμβανόμενης Τοπικής Αναζήτησης

Η τεχνική της επαναλαμβανόμενης τοπικής αναζήτησης (Iterated Local Search) [9, 59] επεκτείνει την τεχνική της τοπικής αναζήτησης βασιζόμενη στο γεγονός ότι η δεύτερη παγιδεύεται σε τοπικά βέλτιστες λύσεις. Αντίθετα, η επαναλαμβανόμενη



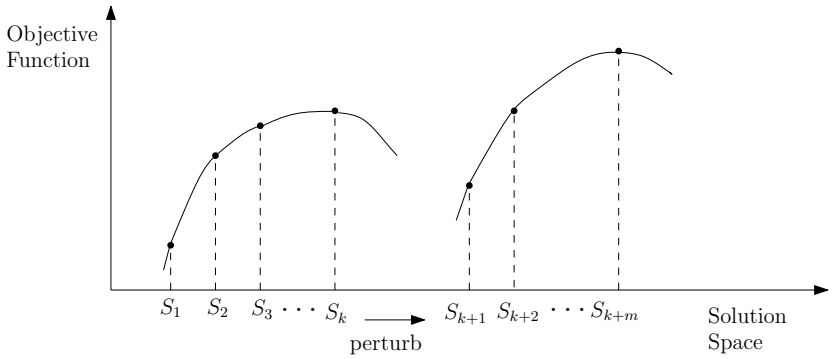
Εικόνα 3.1: Παράδειγμα μιας εφικτής λύσης για το TOPTW

τοπική αναζήτηση μπορεί να ξεφύγει με ένα κατάλληλο βήμα διαταραχής. Στην επαναλαμβανόμενη τοπική αναζήτηση αρχικά κατασκευάζεται μια εφικτή λύση του προβλήματος και εφαρμόζεται τοπική αναζήτηση μέχρι να βρεθεί μια τοπικά βέλτιστη λύση S^*. Ακολούθως, έως ότου ικανοποιηθεί μια συνθήκη (που συνήθως είναι ένα πλήθος επαναλήψεων, ένα χρονικό όριο εκτέλεσης ή ένα πλήθος επαναλήψεων χωρίς να έχει προκύψει καλύτερη λύση), εκτελείται ένας βρόγχος. Στον βρόγχο αρχικά παράγεται μια καινούργια εφικτή λύση S' (ξεφεύγοντας με αυτόν τον τρόπο από το τοπικό βέλτιστο), μέσω ενός βήματος διαταραχής. Το βήμα διαταραχής παίρνει υπ' όψιν του εκτός από την προηγούμενη λύση και την ιστορία της όλης διαδικασίας. Έπειτα εφαρμόζεται τοπική αναζήτηση στην S' έως ότου προκύψει ένα νέο τοπικό βέλτιστο S''. Στο τέλος του βρόγχου, η νέα λύση γίνεται αποδεκτή με βάση ένα επιλεγμένο κριτήριο (που συνήθως είναι αν η καινούργια λύση δίνει καλύτερη τιμή στην συνάρτηση χρησιμότητας από την προηγούμενη). Η γενική ιδέα της επαναλαμβανόμενης τοπικής αναζήτησης συνοψίζεται στον Αλγόριθμο 3.1.

Αλγόριθμος 3.1 Η τεχνική της επαναλαμβανόμενης τοπικής αναζήτησης

1: Κατασκεύασε μια αρχική λύση S
2: Χρησιμοποίησε τοπική αναζήτηση για να πάρεις μια τοπικά βέλτιστη λύση S^*
3: **Ενόσω** η συνθήκη δεν έχει ικανοποιηθεί **Κάνε**
4: Διατάραξε την S^* με βάση την ιστορία της διαδικασίας παίρνοντας την λύση S'
5: Εφάρμοσε τοπική αναζήτηση στην S' για να βρεις την λύση S''
6: Διάλεξε με βάση κάποιο κριτήριο μία από τις S^*, S'' ως την νέα λύση S^*
7: **Τέλος Ενόσω**
Επέστρεψε S^*

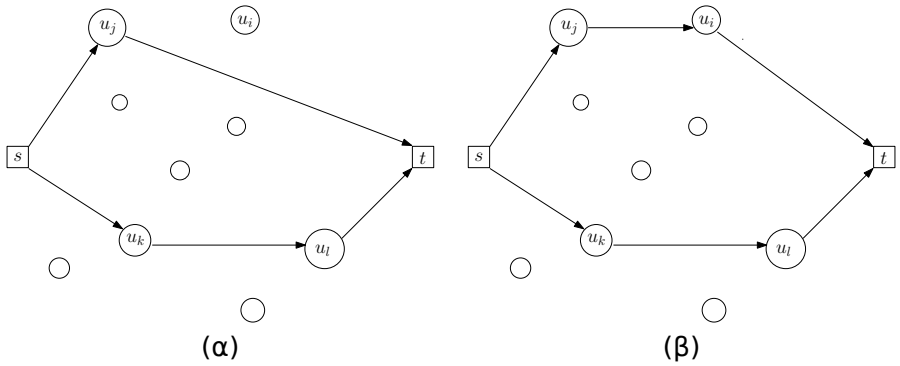
Η Εικόνα 3.2 παρουσιάζει ένα τυπικό παράδειγμα της εκτέλεσης της Επαναλαμβανόμενης Τοπικής Αναζήτησης. Δεδομένης μιας αρχικής λύσης S_1, η τοπική αναζήτηση καταλήγει στο τοπικό βέλτιστο S_k. Αντίθετα, στην Επαναλαμβανόμενη Τοπική Αναζήτησης, το τοπικό αυτό βέλτιστο διαταράσσεται φτάνοντας με αυτόν τον τρόπο στην λύση S_{k+1}. Τότε, εκτελώντας πάλι τοπική αναζήτηση, βρίσκεται ένα καλύτερο τοπικό βέλτιστο, το S_{k+m}.



Εικόνα 3.2: Παράδειγμα της εκτέλεσης της Επαναλαμβανόμενης Τοπικής Αναζήτησης

3.3 Ο ILS αλγόριθμος των Vansteenwegen et al.

Ο αλγόριθμος των Vansteenwegen et al. [80] είναι ένας μεταευρετικός αλγόριθμος βασιζόμενος στην τεχνική της επαναλαμβανόμενης τοπικής αναζήτησης. Η γειτονιά που ορίζεται για την τοπική αναζήτηση είναι η εξής: Μια εφικτή λύση S' του TOPTW είναι γειτονική μιας εφικτής λύσης S αν και μόνο αν η S' προκύπτει από την S με ένα βήμα εισαγωγής, εισάγοντας έναν κόμβο σε ένα μονοπάτι της S. Για παράδειγμα, στην Εικόνα 3.3(β) παρουσιάζεται μια γειτονική λύση της λύσης που απεικονίζεται στην Εικόνα 3.3(α).



Εικόνα 3.3: Η (β) αποτελεί μια γειτονική λύση της (α)

Για να εκτελείται το βήμα εισαγωγής αποδοτικά κάθε κόμβος u_i που ανήκει σε μία διαδρομή r_m έχει τις εξής επιπλέον πληροφορίες: (i) τον χρόνο άφιξης arrive_i, (ii) τον χρόνο έναρξης της επίσκεψης start_i, (iii) τον χρόνο αναμονής μέχρι να ξεκινήσει η επίσκεψη wait_i, (iv) τον μέγιστο χρόνο που μπορεί να παραταθεί η έναρξη της επίσκεψης στον κόμβο maxShift_i και η διαδρομή να ικανοποιεί τους χρονικούς της περιορισμούς.

Για κάθε διαδρομή r_m και κόμβο u_i που ανήκει σε αυτήν, αν next_i/prev_i είναι ο δείκτης του κόμβου που έπεται/προηγείται του u_i, τότε θα πρέπει να ισχύουν τα εξής:

$\text{start}_1 = 0$ και

- $\text{wait}_i = \max\{0, R_{im} - \text{arrive}_i\}$
- $\text{start}_i = \text{arrive}_i + \text{wait}_i$
- $\text{arrive}_{\text{next}_i} = \text{start}_i + v_i + c_{i,\text{next}_i}$

ενώ για το maxShift πρέπει να ισχύει ότι $\text{maxShift}_N = B_m - \text{arrive}_N$ και $\text{maxShift}_{\text{prev}_i} = \min\{D_{\text{prev}_i,m} - \text{start}_{\text{prev}_i}, \text{wait}_i + \text{maxShift}_i\}$.

Έστω η διαδρομή r_m και u_j, u_k δύο διαδοχικοί κόμβοι της, οπότε $k = \text{next}_j$. Αν ο κόμβος u_i εισέλθει μεταξύ των u_j και u_k, ο επιπλέον χρόνος που θα καταναλωθεί στην διαδρομή θα είναι ίσος με $\text{shift}_i^j = c_{ji} + \text{wait}_i^j + v_i + c_{ik} - c_{jk}$, όπου wait_i^j θα είναι ο χρόνος αναμονής στον κόμβο u_i μέχρι να ξεκινήσει η επίσκεψη σε αυτόν. Η εισαγωγή αυτή είναι εφικτή αν και μόνο $\text{shift}_i^j \leq \text{wait}_k + \text{maxShift}_k$ και $\text{start}_j + v_j + c_{ji} \leq D_{im}$.

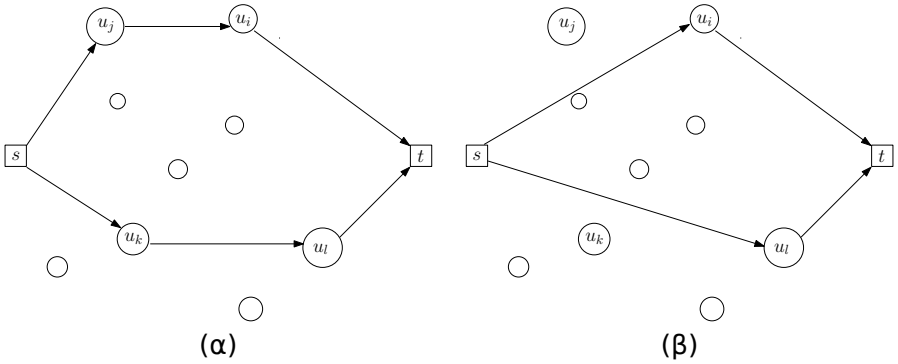
Στο βήμα εισαγωγής (**ILS_Insert**) εξετάζονται όλοι οι κόμβοι που δεν ανήκουν σε κάποια διαδρομή της λύσης. Για κάθε μη συμπεριλαμβανόμενο κόμβο u_i εξετάζονται όλες οι πιθανές θέσεις εισαγωγής στην λύση, δηλαδή όλοι οι κόμβοι κάθε διαδρομής εκτός από τον u_N, και υπολογίζεται η εφικτή θέση (αν υπάρχει) στην οποία ο u_i έχει τον ελάχιστο χρόνο κατανάλωσης (shift). Ο ελάχιστος χρόνος κατανάλωσης θα θεωρείται ως το shift_i. Τότε, θεωρείται ένας λόγος (ratio) που εκφράζει πόσο κερδοφόρα είναι η εισαγωγή ενός κόμβου, σε συνδυασμό με τον χρόνο κατανάλωσης στην διαδρομή. Ο λόγος υπολογίζεται ως $\text{ratio}_i = \frac{p_i^2}{\text{shift}_i}$. Στο τέλος του βήματος ο κόμβος u_i με τον μεγαλύτερο λόγο εισάγεται στην θέση που ελαχιστοποιείται ο χρόνος κατανάλωσης του. Ύστερα από την εισαγωγή ενημερώνονται οι χρονικές μεταβλητές arrive, start, wait και maxShift του u_i και των κόμβων που έπονται του u_i. Επιπλέον, ενημερώνονται και οι χρόνοι maxShift των κόμβων που προηγούνται του u_i. Ο ψευδοκώδικας του βήματος ILS_instert δίνεται στον Αλγόριθμο 3.2.

Αλγόριθμος 3.2 Το βήμα εισαγωγής του ILS

Για κάθε κόμβο που δεν ανήκει στην λύση **Κάνε**
  Βρες την θέση με το μικρότερο shift
  Υπολόγισε το ratio
Τέλος Για
Εισήγαγε τον κόμβο με το μεγαλύτερο ratio στην θέση με το μικρότερο shift
Ενημέρωσε τις τιμές των arrive, start, wait και maxShift στην διαδρομή που έγινε η εισαγωγή

Στο βήμα της διαταραχής (**Shake**) αφαιρείται ένα πλήθος διαδοχικών κόμβων από κάθε διαδρομή. Πιο συγκεκριμένα, το Shake παίρνει σαν παραμέτρους δύο ακεραίους removeNumber και startNumber. Η παράμετρος removeNumber δηλώνει το πλήθος των διαδοχικών κόμβων που θα αφαιρεθούν από κάθε διαδρομή, ενώ η startNumber δηλώνει την θέση του αρχικού κόμβου που θα αφαιρεθεί. Αν σε κάποια στιγμή της διαδικασίας, είναι να αφαιρεθεί ο τερματικός κόμβος κάποιας διαδρομής, η αφαίρεση των κόμβων συνεχίζεται από τον κόμβο που έπεται του αρχικού κόμβου της διαδρομής. Ένα παράδειγμα του Shake με παραμέτρους removeNumber και startNumber ίσα με 1 δίνεται στην Εικόνα 3.4.

Η περιγραφή του ILS αλγορίθμου ακολουθεί. Τα startNumber και removeNumber αρχικοποιούνται στην μονάδα, ενώ το μέγιστο επιτρεπόμενο πλήθος κόμβων που μπορούν να αφαιρεθούν από μία διαδρομή (maxRem) τίθεται ίσο με $\frac{N}{3K}$. Ο αλγόριθμος εκτελεί έναν βρόγχο εώς ότου δεν βρεθεί καλύτερη λύση, δηλαδή λύση με μεγαλύτερο κέρδος από την βέλτιστη ως εκείνη την στιγμή, για $\text{maxIterations} = 150$ διαδοχικές επαναλήψεις. Μέσα στον βρόγχο, αρχικά παράγεται μια τοπικά βέλτιστη λύση εφαρμόζοντας επαναληπτικά το βήμα εισαγωγής. Αν αυτή η λύση είναι η πιο κερδοφόρα που έχει βρεθεί, τότε καταγράφεται στην μνήμη και τα removeNumber, notImproved (το πλήθος των διαδοχικών επαναλήψεων του βρόγχου χωρίς βελτίωση της λύσης) γίνονται 1 και 0, αντίστοιχα. Αλλιώς το notImproved αυξάνεται κατά ένα. Ύστερα, το βήμα διαταραχής εφαρμόζεται στην τωρινή λύση. Τέλος, οι τιμές startNumber και removeNumber ενημερώνονται, με το startNumber να αυξάνεται κατά removeNumber και το removeNumber να αυξάνεται κατά 1. Αυτή η ενημέρωση γίνεται με σκοπό να



Εικόνα 3.4: Πριν το Shake (α) και μετά το Shake (β)

εκτελείται το βήμα διαταραχής με διαφορετικές παραμέτρους κάθε φορά ώστε να εξερευνηθεί καλύτερα ο χώρος των λύσεων. Τέλος, αν το startNumber ξεπεράσει το μέγεθος της μικρότερης διαδρομής, τότε μειώνεται όσο είναι το μέγεθος αυτό, και επίσης, όταν το removeNumber φτάσει το μέγιστο επιτρεπόμενο πλήθος αφαιρούμενων κόμβων τότε γίνεται ίσο με 1. Ο ψευδοκώδικας του ILS αλγορίθμου δίνεται στον Αλγόριθμο 3.3.

Αλγόριθμος 3.3 Ο **ILS** αλγόριθμος των Vansteenwegen et al. [80]

bestSolution $\leftarrow$ null
bestProfit $\leftarrow 0$
Αρχικοποίησε την currentSolution ως το σύνολο των K διαδρομών που αρχίζουν στον κόμβο u_1 και τερματίζουν στον κόμβο u_N.
maxIterations $\leftarrow 150$
maxRem $\leftarrow \frac{N}{3K}$
notImproved $\leftarrow 0$
startNumber $\leftarrow 1$
removeNumber $\leftarrow 1$
Ενόσω notImproved $<$ maxIterations **Κάνε**
  Ενόσω η currentSolution δεν είναι τοπικά βέλτιστη **Κάνε**
    ILS_Insert
  Τέλος Ενόσω
  currentProfit $\leftarrow$ το κέρδος της currentSolution
  Αν currentProfit $>$ bestProfit **Τότε**
    bestSolution $\leftarrow$ currentSolution
    bestProfit $\leftarrow$ currentProfit
    removeNumber $\leftarrow 1$
    notImproved $\leftarrow 0$
  Αλλιώς
    notImproved $\leftarrow$ notImproved $+ 1$
  Τέλος Αν
  Shake(removeNumber,startNumber)
  startNumber $\leftarrow$ startNumber $+$ removeNumber
  removeNumber $\leftarrow$ removeNumber $+ 1$
  smallestSize $\leftarrow$ το μέγεθος της μικρότερης διαδρομής στην λύση
  Αν startNumber $\geq$ smallestSize **Τότε**
    startNumber $\leftarrow$ startNumber $-$ smallestSize
  Τέλος Αν
  Αν removeNumber = maxRem **Τότε**
    removeNumber $\leftarrow 1$
  Τέλος Αν
Τέλος Ενόσω
Επέστρεψε bestSolution

Ο ILS αλγόριθμος έχει λίγο χρόνο εκτέλεσης, παράγοντας ταυτόχρονα λύσεις με υψηλό κέρδος. Όπως αναφέρεται στο [80], ο αλγόριθμος συγκρίνεται με τον αλγόριθμο των Montemanni και Gambardella [62] στα στιγμιότυπα που έχουν δημιουργήσει οι δεύτεροι. Τα στιγμιότυπα επεκτείνουν εκείνα που δημιούργησαν ο Solomon [73] και οι Cordeau et al. [22] για το OPTW. Τα αποτελέσματα του ILS υπολείπονται στο κέρδος κατά μέσο όρο λιγότερο από 2% από τον αλγόριθμο των Montemanni και Gambardella, ενώ ο χρόνος εκτέλεσης του ILS είναι εκατοντάδες φορές γρηγορότερος.

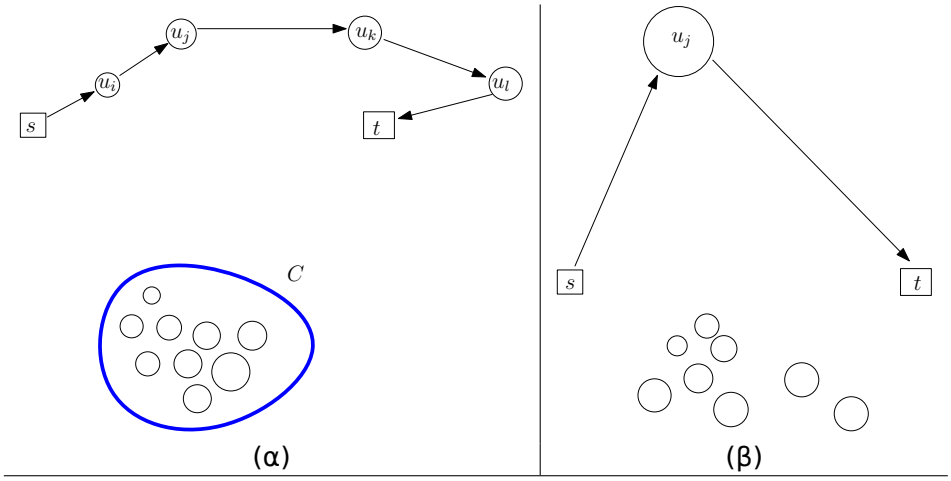
Τα μειονεκτήματα του ILS έγκεινται στο γεγονός ότι ο αλγόριθμος κατά την εισαγωγή των κόμβων παίρνει υπ' όψιν του μόνο το κέρδος που θα συνεισφέρει ο εισηγμένος κόμβος και τον χρόνο που θα καταναλωθεί λόγω αυτής της εισαγωγής. Αυτό οδηγεί στα επόμενα μειονεκτήματα:

- Ο αλγόριθμος μπορεί να μην εισάγει κόμβους που να απέχουν μεγάλη απόσταση από την αρχή και το τέλος των διαδρομών παρ' όλο που έχουν αρκετά μεγάλο κέρδος. Αυτό γίνεται ακόμα πιο εμφανές αν υπάρχει μια μακρινή, από την αρχή και το τέλος, γειτονιά με αρκετούς κόμβους με μεγάλο κέρδος. Η λύση μπορεί να μην συμπεριλάβει κανέναν από αυτούς τους κόμβους γιατί σε κάθε βήμα εισαγωγής θα έχουν μικρό λόγο λόγω της μεγάλης απόστασής τους από τα άκρα των διαδρομών. Παρ' όλα αυτά αν κάποιος από τους κόμβους της γειτονιάς εισαγόταν στην λύση κάποια στιγμή, τότε μπορεί ολόκληρη η γειτονιά να συμπεριλαμβανόταν και να έδινε μια πολύ πιο κερδοφόρα λύση από αυτήν που τελικά θα επιστραφεί. Για παράδειγμα στην διαδρομή της Εικόνας 3.5(α) εισάγονται διαδοχικά οι κόμβοι u_i, u_j, u_k, u_l λόγω του μικρού χρονικού κόστους που θα επιβαρύνουν την διαδρομή. Πλέον, οι κόμβοι του συνόλου C δεν μπορούν να εισαχθούν λόγω του χρονικού περιθωρίου της διαδρομής. Παρ' όλα αυτά αν κάποιος από αυτούς είχε εισηχθεί, τότε αρκετοί ακόμα θα είχαν εισηχθεί λόγω της κοντινής μεταξύ τους απόστασης, οδηγώντας στην παραγωγή μια πιο κερδοφόρας λύσης.
- Αντίστροφα, μπορεί να υπάρχουν κάποιοι κόμβοι με πολύ μεγάλο κέρδος, απομακρυσμένοι από τους υπόλοιπους, που θα συμπεριληφθούν στην λύση λόγω του μεγάλου κέρδους τους. Αυτό θα έχει ως αποτέλεσμα η λύση να παγιδευτεί σε αυτούς, δηλαδή να μην μπορούν να εισαχθούν άλλοι κόμβοι λόγω των χρονικών περιθωρίων των διαδρομών. Οπότε, παρ' όλο που μπορεί να υπάρχει μια γειτονιά με πολλούς κόμβους με μέτριο κέρδος, κανένας από αυτούς τους κόμβους δεν θα εισαχθεί ποτέ στην λύση. Ενώ αν κάποιοι από τους κόμβους της γειτονιάς εισάγονταν, μια πιο κερδοφόρα λύση θα παραγόταν. Για παράδειγμα στην Εικόνα 3.5(β) παρουσιάζεται μια διαδρομή που έχει παγιδευτεί στον κόμβο u_j και δεν μπορεί να επισκεφτεί κανέναν άλλο κόμβο.

3.4 Προτεινόμενοι αλγόριθμοι

Σε αυτήν την ενότητα, προτείνονται οι αλγόριθμοι **CSCRatio** (Cluster Search Cluster Ratio) και **CSCRoutes** (Cluster Search Cluster Routes). Οι αλγόριθμοι βασίζονται στον ILS και επιχειρούν να βελτιώσουν τις προαναφερθείσες αδυναμίες του.

Σε ένα προπαρασκευαστικό στάδιο, οι κόμβοι κάθε τοπολογίας διαμερίζονται σε clusters (ομάδες) με βάση την μεταξύ τους απόστασή, εφαρμόζοντας τον αλγόριθμο global k-means [4, 57]. Ο αριθμός των clusters (M) που θα δημιουργηθούν, δίνεται ως παράμετρος στον αλγόριθμο. Ο global k-means διαμερίζει τους κόμβους σε M μη τεμνόμενα ανά δύο clusters, που καλύπτουν το σύνολο των κόμβων, χρησιμοποιώντας ως υπορουτίνα τον αλγόριθμο k-means [60]. Σε κάθε ένα από τα clusters $C_i, i = 1, 2, \ldots, M$



Εικόνα 3.5: Μειονεκτήματα του ILS, το μέγεθος των κόμβων είναι ανάλογο του κέρδους τους

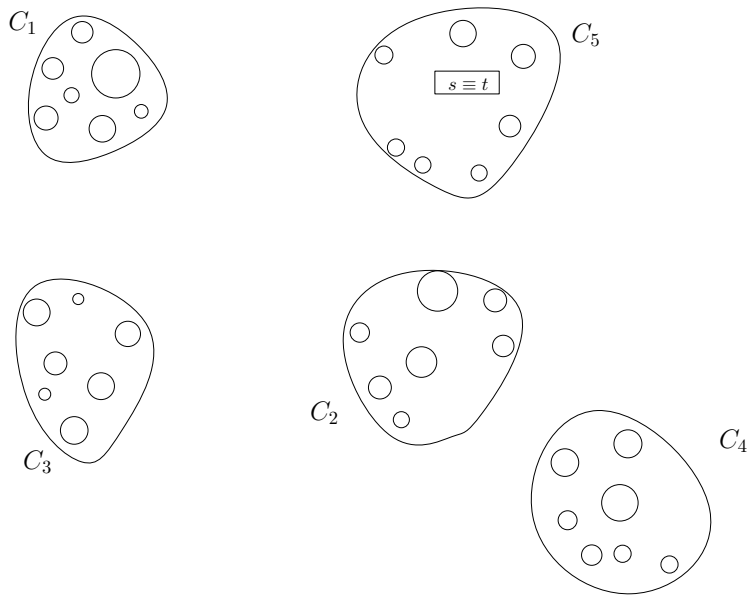
αντιστοιχίζεται ένα κέντρο. Ο κόμβος $u \in V$ ανήκει στο cluster C_i αν και μόνο αν η απόσταση του από το κέντρο του C_i είναι μικρότερη από την απόσταση από κάθε άλλο κέντρο ενός cluster.

Ο k-means είναι ένας αλγόριθμος τοπικής αναζήτησης που δοθέντος (μπορεί επίσης να παραχθεί είτε αυθαίρετα είτε με κάποιο κριτήριο) ενός αρχικού συνόλου από Μ κέντρα clusters εκτελεί επαναληπτικά την εξής διαδικασία: (i) Εισάγει κάθε κόμβο στο πιο κοντινό του cluster (στο cluster με το πιο κοντινό σε αυτό κέντρο). (ii) Υπολογίζει τα νέα κέντρα των clusters ως τον γεωμετρικό μέσο των κόμβων που ανήκουν στα clusters. Η διαδικασία αυτή σταματά όταν δεν υπάρχει κόμβος που να αλλάζει το cluster που ανήκει. Το κυρίαρχο μειονέκτημα του k-means είναι ότι εξαρτάται σε πολύ μεγάλο βαθμό από την επιλογή των αρχικών κεντρών των clusters [68].

Ο global k-means αντίθετα, είναι ένας αιτιοκρατικός αλγόριθμος ανεξάρτητος από οποιαδήποτε αρχική κατάσταση. Ο αλγόριθμος παράγει την λύση με βάση την λύση των υποπροβλημάτων για l=1,2,...,M-1 clusters. Για l=1 το κέντρο του μοναδικού cluster είναι ο γεωμετρικός μέσος των κόμβων. Με βάση την βέλτιστη τοποθέτηση των l κεντρών clusters, η τοποθέτηση των l+1 παράγεται, εφαρμόζοντας $|V|$ φορές τον αλγόριθμο k-means με αρχικά κέντρα τα l που είχαν προηγουμένως βρεθεί και επιπρόσθετα έναν κόμβο του γράφου. Η τοποθέτηση των l+1 θα είναι εκείνη που θα έχει το ελάχιστο άθροισμα τετραγώνων των αποστάσεων των κόμβων από τα κέντρα των clusters που ανήκουν. Ένα παράδειγμα της ομαδοποίησης μιας τοπολογίας σε 5 clusters δίνεται στην Εικόνα 3.6.

Το βήμα εισαγωγής των προτεινόμενων αλγορίθμων επεκτείνει το βήμα εισαγωγής του ILS παίρνοντας υπ' όψιν και το cluster που ανήκει κάθε κόμβος. Επιπλέον, οι αλγόριθμοι φροντίζουν να επισκεφτούν και απομακρυσμένα clusters ώστε να μπορεί να ερευνηθεί καλύτερα ο χώρος των λύσεων. Θεωρώντας ότι πολλά προβλήματα της πραγματικότητας μοντελοποιούν πόλεις σαν γράφους, με τα αξιοθέατα να είναι οι κόμβοι του γράφου, μια ομαδοποίηση των κόμβων μπορεί να υπάρχει ήδη, π.χ. η ομαδοποίηση των αξιοθέατων με βάση τους δήμους που υπάγονται ή αν είναι στα ανατολικά, βόρεια, δυτικά ή νότια προάστια της πόλης.

Οι διαδρομές αρχικοποιούνται και στους δυο αλγορίθμους μέσω ενός κοινού βήματος, του RouteInitPhase. Στο RouteInitPhase δίνεται ως είσοδος ένα σύνολο από K διαφορετικά ανά δύο clusters $C_1, C_2, \ldots, C_K$ και με βάση αυτά, σε κάθε μονοπάτι

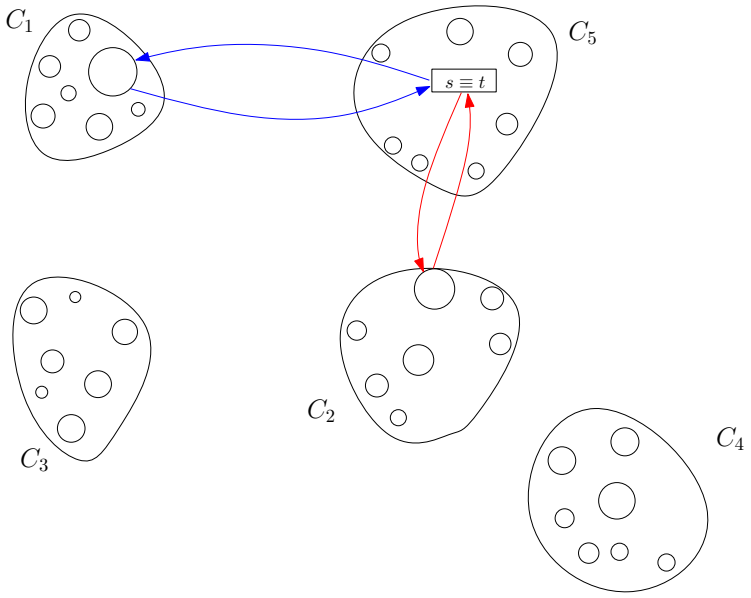


Εικόνα 3.6: Παράδειγμα ομαδοποίησης των κόμβων σε 5 clusters

$r_j, j = 1, \ldots, K$ εισάγεται ο κόμβος u_i από το C_j με τον μεγαλύτερο λόγο $\frac{p_i^2}{\text{shift}_i}$. Μια λίστα από $K-$άδες clusters, listOfClusterSets παράγεται σε κάθε έναν από τους δύο αλγορίθμους, σε ένα προπαρασκευαστικό στάδιο, ώστε να μπορούν να εκτελεστούν διαδοχικά τα RouteInitPhase. Η επιλογή των στοιχείων της λίστας μπορεί να γίνει με διαφορετικούς τρόπους είτε παίρνοντας υπ' όψιν το συνολικό κέρδος των κόμβων των clusters, την απόσταση τους από την αρχή και το τέλος των διαδρομών, την απόσταση των κόμβων μέσα στο cluster ή μπορεί να χρησιμοποιηθεί μια τυχαία διαμέριση των clusters σε σύνολα με K στοιχεία. Στους αλγορίθμους χρησιμοποιείται ο τελευταίος τρόπος. Η αρχικοποίηση των διαδρομών με κόμβους από διαφορετικά clusters ευνοεί την εξερεύνηση απομακρυσμένων περιοχών ενώ ταυτόχρονα δεν παγιδεύει την λύση σε απομονωμένους κόμβους. Ένα παράδειγμα του βήματος RouteInitPhase με είσοδο τα C_1, C_2 δίνεται στην Εικόνα 3.7.

3.4.1 Ο αλγόριθμος CSCRatio

Ο αλγόριθμος CSCRatio είναι σχεδιασμένος με σκοπό να ευνοεί την εισαγωγή κόμβων πριν ή μετά από κόμβους του ίδιου cluster. Το βήμα εισαγωγής του αλγορίθμου **CSCRatio_Insert** επεκτείνει το βήμα εισαγωγής του ILS θεωρώντας μια επιπλέον ποσότητα που δέχεται ως παράμετρο, την clusterParameter. Όσο πιο μεγάλη είναι η παράμετρος τόσο πιο πολύ ευνοείται η εισαγωγή ενός κόμβου κοντά σε κόμβους του ίδιου cluster. Πιο συγκεκριμένα, θεωρείται το shiftCluster μια γενίκευση του shift που ορίστηκε στον ILS και ορίζεται ως εξής: η εισαγωγή του κόμβου u_i έπειτα από τον κόμβο u_j έχει $\text{shiftCluster}_i^j = \frac{\text{shift}_i^j}{\text{clusterParameter}}$ αν είτε ο u_j είτε ο επόμενος του στο μονοπάτι ανήκουν στο ίδιο cluster με τον u_i. Αλλιώς το shiftCluster_i^j είναι ίσο με το shift_i^j. Η πιθανή θέση εισαγωγής του u_j είναι η θέση με το μικρότερο shiftCluster (shiftCluster_i). Αφού βρεθεί αυτή η θέση, υπολογίζεται ο λόγος $\text{ratio}_i = \frac{p_i^2}{\text{shiftCluster}_i}$ που είναι το μέτρο σύγκρισης μεταξύ των υποψήφιων κόμβων για εισαγωγή. Στο τέλος του βήματος



Εικόνα 3.7: Παράδειγμα του RouteInitPhase με είσοδο τα C_1, C_2

ο κόμβος με τον μεγαλύτερο λόγο εισάγεται στην θέση με το μικρότερο shiftCluster.

Στην αρχή του αλγορίθμου ευνοείται η ύπαρξη διαδοχικών κόμβων από το ίδιο cluster στα μονοπάτια, ορίζοντας το clusterParameter ίσο με 1.3. Κατά την διάρκεια της εκτέλεσης του αλγορίθμου η τιμή μειώνεται σταδιακά κατά 0.1 κάθε φορά που έχουν εκτελεστεί $\frac{\text{maxIterations}}{4}$ (όπου maxIterations το μέγιστο πλήθος συνεχόμενων επαναλήψεων χωρίς βελτίωση της λύσης) συνεχόμενες επαναλήψεις χωρίς βελτίωση της λύσης, μέχρι να γίνει 1. Στο τελευταίο τέταρτο των επαναλήψεων το βήμα εισαγωγής πλέον ταυτίζεται με το βήμα εισαγωγής του ILS. Με αυτήν την διακύμανση στις τιμές του clusterParameter επιτυγχάνεται διαφοροποίηση στον τρόπο εισαγωγής κατά την διάρκεια του αλγορίθμου. Αυτό έχει ως αποτέλεσμα την καλύτερη εξερεύνηση του συνόλου των λύσεων.

Το βήμα διαταραχής του αλγορίθμου είναι ίδιο με το βήμα διαταραχής του ILS, όμως πλέον το πλήθος των κόμβων που επιτρέπεται να διαγραφούν δεν είναι ίσο με $\frac{N}{3K}$ αλλά ίσο με το μισό του μεγέθους της διαδρομής της λύσης με το μέγιστο πλήθος κόμβων. Με αυτό τον τρόπο δεν αφαιρείται μεγάλο μέρος της λύσης. Οπότε, ένα τοπικό βέλτιστο μπορεί να ξαναβρεθεί με μικρό αριθμό βημάτων εισαγωγής, εξοικονομώντας χρόνο εκτέλεσης. Άρα, το πλήθος των επαναλήψεων του αλγορίθμου μπορεί να είναι μεγαλύτερο από του ILS χωρίς να έχει μεγαλύτερο χρόνο εκτέλεσης.

Χρόνος εκτέλεσης εξοικονομείται επίσης στο βήμα εισαγωγής του αλγορίθμου διαγράφοντας από την λίστα των κόμβων που είναι υποψήφιοι για εισαγωγή εκείνους που κατά την διάρκεια του βήματος δεν μπορούσαν να εισαχθούν σε καμιά διαδρομή λόγω των χρονικών περιορισμών. Αυτοί οι κόμβοι εξετάζονται πάλι για εισαγωγή μετά την εφαρμογή του βήματος διαταραχής.

Ο αλγόριθμος CSCRatio δέχεται δύο παράμετρους, τους αριθμούς numberOfClusters και maxIterations. Ο numberOfClusters δηλώνει το πλήθος των clusters που πρέπει να δημιουργηθούν, ενώ ο maxIterations δηλώνει το πλήθος των επιτρεπόμενων επαναλήψεων της τοπικής αναζήτησης χωρίς εύρεση καλύτερης λύσης. Στο προπαρασκευαστικό στάδιο, κατασκευάζονται numberOfClusters clusters χρησιμοποιώντας τον global k-means όπως επίσης και η λίστα listOfClusterSets.

Η online διαδικασία του αλγορίθμου εκτελείται έως ότου η λίστα listOfClusterSets αδειάσει. Μέσα στον βρόγχο πρώτα αφαιρείται η πρώτη $K-$άδα από clusters της λίστας και δίνεται ως όρισμα στην διαδικασία RouteInitPhase. Τότε, οι μεταβλητές notImproved, startNumber, removeNumber (όμοιες με αυτές που χρησιμοποιήθηκαν στον ILS) γίνονται ίσες με $0, 1$ και 1, αντίστοιχα. Ύστερα, ένας εσωτερικός βρόγχος εκτελείται όσο το πλήθος των επαναλήψεων χωρίς βελτίωση της λύσης είναι μικρότερο από maxIterations. Μέσα στον βρόγχο εκτελούνται τα ακόλουθα: (i) Επιλέγεται με βάση το πλήθος των επαναλήψεων χωρίς βελτίωση η τιμή της παραμέτρου clusterParameter. Η τιμή αρχικά ισούται με 1.3 και φθίνει κατά 0.1 κάθε ένα τέταρτο των maxIterations. (ii) Εφαρμόζεται η τοπική αναζήτηση με το βήμα εισαγωγής να εκτελείται έως ότου μια τοπικά βέλτιστη λύση βρεθεί (δεν μπορεί να προστεθεί κανένας άλλος κόμβος στην λύση). Αν η τοπικά βέλτιστη λύση που βρεθεί είναι η καλύτερη που έχει βρεθεί στην διαδικασία, καταγράφεται και οι τιμές των removeNumber και notImproved γίνονται 1 και 0 αντίστοιχα, αλλιώς το notImproved αυξάνεται κατά 1. (iii) Εκτελείται το βήμα διαταραχής, με τον περιορισμό ότι το removeNumber δεν ξεπερνάει το μισό του μεγέθους της μέγιστης διαδρομής. (iv) Οι τιμές των startNumber και removeNumber ενημερώνονται παρόμοια με τον ILS. Όταν το πλήθος των επαναλήψεων χωρίς βελτίωση γίνει ίσο με maxIterations, τότε όλοι οι κόμβοι εκτός από την αρχή και το τέλος κάθε διαδρομής αφαιρούνται από την τρέχουσα λύση. Ο ψευδοκώδικας του αλγορίθμου δίνεται στον Αλγόριθμο 3.4.

3.4.2 Ο αλγόριθμος CSCRoutes

Ο αλγόριθμος CSCRoutes είναι σχεδιασμένος με σκοπό να είναι γρήγορος και να κατασκευάζει διαδρομές χωρίς πολλές μετακινήσεις ανάμεσα στα clusters. Πιο συγκεκριμένα, αν κάποια διαδρομή επισκέπτεται κάποιο cluster, ο αλγόριθμος απαγορεύει την αποχώρηση από το cluster και μετά την επιστροφή πάλι σε αυτό.

Δεδομένης μίας διαδρομής r της λύσης κάθε μεγιστική ακολουθία διαδοχικών κόμβων στην διαδρομή από το ίδιο cluster C καλείται CR (cluster route) του cluster C της r και συμβολίζεται με CR^r_C. Σε κάθε διαδρομή επιτρέπουμε να υπάρχει το πολύ ένα CR από κάθε cluster με την εξαίρεση του cluster C_d που περιέχει την αρχή και το τέλος της διαδρομής αν αυτά ανήκουν στο ίδιο cluster. Τότε, επιτρέπουμε να υπάρχουν δύο CR του cluster C_d όπου το πρώτο CR^r_f θα ξεκινάει από την αρχή και το δεύτερο CR^r_l θα τελειώνει στο τέλος της διαδρομής. Για παράδειγμα στην Εικόνα 3.8(α) παρουσιάζεται μια λύση που μπορεί να προκύψει από τον CSCRoutes, ενώ η λύση που δίνεται στην Εικόνα 3.8(β) δεν μπορεί να προκύψει καθώς η κόκκινη διαδρομή επισκέπτεται το cluster C_2, αποχωρεί από αυτό για το cluster C_4 και επιστρέφει ξανά σε αυτό.

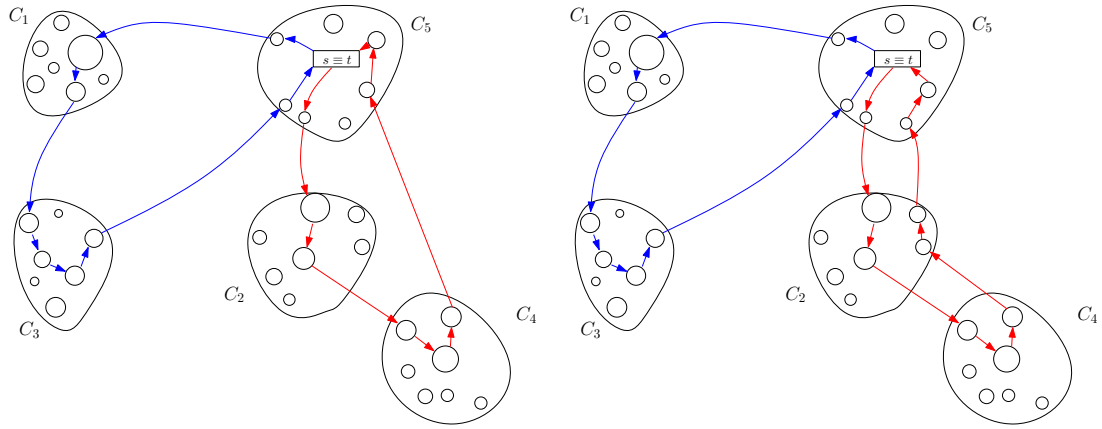
Το βήμα εισαγωγής **CSCRoutes_Insert** του CSCRoutes εισάγει τον κόμβο u_i με τον μεγαλύτερο λόγο $\frac{p_i^2}{\mathsf{shift}_i}$, μη επιτρέποντας όμως εισαγωγές που θα δημιουργούσαν δύο CR ενός cluster σε μία διαδρομή. Αφού ένας κόμβος δεν μπορεί να εισέλθει σε κάθε σημείο στην διαδρομή πρέπει να διακριθούν οι θέσεις όπου μπορεί να συμβεί η εισαγωγή. Επίσης, όπως και στον CSCRatio αν ένας κόμβος βρεθεί ότι δεν μπορεί να εισέλθει σε καμιά διαδρομή τότε δεν εξετάζεται για εισαγωγή κατά την διάρκεια της τοπικής αναζήτησης και θεωρείται ξανά για εισαγωγή μετά από την εφαρμογή του βήματος διαταραχής.

Από εδώ και στο εξής θεωρούμε ότι ο αρχικός (u_1) και τελικός (u_N) κόμβος κάθε διαδρομής ταυτίζονται και θα τον συμβολίζουμε με depot. Επίσης, το cluster κάθε κόμβου u θα συμβολίζεται με cluster(u). Τέλος, θεωρούμε για κάθε διαδρομή r μία λίστα listOfClusters(r) που περιέχει όλα τα clusters που επισκέπτεται η διαδρομή. Το cluster(depot) ανήκει πάντα στην listOfClusters(r) και αν το μέγεθος της λίστας είναι

Αλγόριθμος 3.4 Ο αλγόριθμος **CSCRatio**(numberOfClusters,maxIterations)

Προπαρασκευαστικό Στάδιο
εκτέλεσε τον αλγόριθμο global k-means με παράμετρο numberOfClusters
Δημιούργησε την λίστα listOfClusterSets

Online Διαδικασία
bestSolution $\leftarrow$ null
bestProfit $\leftarrow 0$
Αρχικοποίησε την currentSolution ως το σύνολο των K διαδρομών που αρχίζουν στον κόμβο u_1 και τερματίζουν στον κόμβο u_N.
it1 $\leftarrow \frac{\text{maxIterations}}{4}$
it2 $\leftarrow \frac{2 \cdot \text{maxIterations}}{4}$
it3 $\leftarrow \frac{3 \cdot \text{maxIterations}}{4}$
Ενόσω η listOfClusterSets δεν είναι άδεια **Κάνε**
  ClusterSet $\leftarrow$ listOfClusterSets.pop
  RouteInitPhase(ClusterSet)
  notImproved $\leftarrow 0$
  startNumber $\leftarrow 1$
  removeNumber $\leftarrow 1$
  Ενόσω notImproved $<$ maxIterations **Κάνε**
    Αν notImproved $<$ it2 **Τότε**
      Αν notImproved $<$ it1 **Τότε**
        clusterParameter $\leftarrow 1.3$
      Αλλιώς
        clusterParameter $\leftarrow 1.2$
      Τέλος Αν
    Αλλιώς
      Αν notImproved $<$ it3 **Τότε**
        clusterParameter $\leftarrow 1.1$
      Αλλιώς
        clusterParameter $\leftarrow 1$
      Τέλος Αν
    Τέλος Αν
    Ενόσω η currentSolution δεν είναι τοπικά βέλτιστη **Κάνε**
      CSCRatio_Insert(clusterParameter)
    Τέλος Ενόσω
    currentProfit $\leftarrow$ το κέρδος της currentSolution
    Αν currentProfit $>$ bestProfit **Τότε**
      bestSolution $\leftarrow$ currentSolution
      bestProfit $\leftarrow$ currentProfit
      removeNumber $\leftarrow 1$
      notImproved $\leftarrow 0$
    Αλλιώς
      notImproved $\leftarrow$ notImproved $+ 1$
    Τέλος Αν
    Αν removeNumber $> \frac{\text{μέγεθος της μέγιστης διαδρομής}}{2}$ **Τότε**
      removeNumber $\leftarrow 1$
    Τέλος Αν
    Shake(removeNumber,startNumber)
    startNumber $\leftarrow$ startNumber $+$ removeNumber
    removeNumber $\leftarrow$ removeNumber $+ 1$
    smallestSize $\leftarrow$ το μέγεθος της μικρότερης διαδρομής στην λύση
    Αν startNumber $\geq$ smallestSize **Τότε**
      startNumber $\leftarrow$ startNumber $-$ smallestSize
    Τέλος Αν
  Τέλος Ενόσω
  Αφαίρεσε όλους τους κόμβους της currentSolution εκτός από την αρχή και το τέλος κάθε διαδρομής
Τέλος Ενόσω
Επέστρεψε bestSolution



Εικόνα 3.8: Εφικτή λύση του CSCRoutes (α) και μη εφικτή λύση (β)

1, τότε είναι το μοναδικό cluster που επισκέπτεται η διαδρομή.

Δοθέντος ενός κόμβου u που δεν ανήκει στην λύση και μιας διαδρομής r, στο βήμα εισαγωγής υπάρχουν οι εξής περιπτώσεις:

- cluster(u) = cluster(depot) και η listOfClusters(r) έχει μέγεθος 1. Τότε ο u μπορεί να εισέλθει παντού στην διαδρομή.
- cluster(u) = cluster(depot) και η listOfClusters(r) έχει μέγεθος > 1. Τότε ο u μπορεί να εισέλθει στα CR που σχετίζονται με το depot, δηλαδή στα CR_f^r και CR_l^r.
- cluster(u) $\neq$ cluster(depot) και η listOfClusters(r) έχει μέγεθος 1. Τότε ο u μπορεί να εισέλθει παντού στην διαδρομή. Αν η εισαγωγή πραγματοποιηθεί τότε ένα καινούργιο CR θα δημιουργηθεί με μόνο κόμβο τον u.
- cluster(u) $\neq$ cluster(depot) και η listOfClusters(r) έχει μέγεθος > 1 και περιέχει το cluster(u). Τότε ο u μπορεί να εισέλθει στο CR που σχετίζεται με το cluster του.
- cluster(u) $\neq$ cluster(depot) και η listOfClusters(r) έχει μέγεθος > 1 αλλά δεν περιλαμβάνει το cluster(u). Τότε, ο u μπορεί να εισέλθει μόνο μετά τους κόμβους που είναι τέλος ενός CR εκτός από το depot στο CR_l^r. Αν η εισαγωγή αυτή πραγματοποιηθεί, θα δημιουργηθεί ένα καινούργιο CR με μόνο κόμβο τον u.

Ο ψευδοκώδικας του βήματος CSCRoutes_Insert δίνεται στον Αλγόριθμο 3.5.

Ο CSCRoutes αλγόριθμος είναι παρόμοιος με τον CSCRatio με διαφορά στο βήμα εισαγωγής. Στον CSCRatio το βήμα εισαγωγής επέτρεπε την εισαγωγή νέων κόμβων σε κάθε σημείο μιας διαδρομής, ενώ στον CSCRoutes η εισαγωγή επιτρέπεται μόνο αν δεν δημιουργούνται δύο διαφορετικά cluster routes σε μια διαδρομή. Επιπλέον, το CSCRatio_Insert χρησιμοποιούσε μια παράμετρο clusterParameter, που επηρέαζε τον λόγο ratio, ενώ το CSCRoutes_Insert θεωρεί ως λόγο προτίμησης τον λόγο του κέρδους στο τετράγωνο δια του χρόνου κατανάλωσης.

Για λόγους πληρότητας, παρατίθεται η περιγραφή του CSCRoutes. Ο αλγόριθμος δέχεται τις παραμέτρους numberOfClusters (το πλήθος των clusters που θα δημιουργηθούν) και maxIterations (το πλήθος των επιτρεπόμενων επαναλήψεων χωρίς εύρεση καλύτερης λύσης). Σε ένα προπαρασκευαστικό στάδιο, numberOfClusters clusters δημιουργούνται από τον global k-means αλγόριθμο και δημιουργείται επίσης η λίστα

Αλγόριθμος 3.5 Το βήμα εισαγωγής του CSCRoutes

Για κάθε κόμβο u που δεν ανήκει στην λύση **Κάνε**
clusterID←cluster(u)
Για κάθε διαδρομή r **Κάνε**
Αν clusterID=cluster(depot) **Τότε**
Αν listOfClusters(r) έχει μέγεθος 1 **Τότε**
Υπολόγισε το shift του u μετά από κάθε κόμβο της r.
Αλλιώς
Υπολόγισε το shift του u μετά από κάθε κόμβο στην CR_f^r και πριν από κάθε κόμβο στην CR_l^r.
Τέλος Αν
Αλλιώς ▷ clusterID≠cluster(depot)
Αν listOfClusters(r) έχει μέγεθος 1 **Τότε**
Υπολόγισε το shift του u μετά από κάθε κόμβο της r.
Αλλιώς
Αν listOfClusters(r) δεν περιέχει το clusterID **Τότε**
Υπολόγισε το shift του u μετά από κάθε κόμβο που είναι το τέλος μιας CR.
Αλλιώς
Υπολόγισε το shift του u μετά από κάθε κόμβο στην $CR^r_{\text{cluster}(u)}$ και πριν τον αρχικό της
Τέλος Αν
Τέλος Αν
Τέλος Αν
Τέλος Για
Υπολόγισε τον λόγο του u
Τέλος Για
Εισήγαγε τον κόμβο με τον μέγιστο λόγο
Ανανέωσε τους χρόνους των κόμβων και την λίστα των clusters της διαδρομής που έγινε η εισαγωγή.

listOfClusterSets. Η online διαδικασία εκτελεί έναν βρόγχο έως ότου η listOfClusterSets αδειάσει. Στον βρόγχο, αρχικά αφαιρείται μια $K-$άδα από την λίστα listOfClusterSets, εκτελείται το RouteInitPhase με όρισμα αυτήν την $K-$άδα και αρχικοποιούνται οι τιμές των notImproved (0), startNumber (1) και removeNumber (1). Ύστερα εκτελείται ένας εσωτερικός βρόγχος έως ότου το πλήθος των επαναλήψεων χωρίς βελτίωση (notImproved) της λύσης γίνει ίσο με maxIterations. Στον δεύτερο βρόγχο αρχικά εκτελείται τοπική αναζήτηση χρησιμοποιώντας το βήμα εισαγωγής του αλγορίθμου. Αν η τοπική αναζήτηση βρει μια καλύτερη λύση από την προϋπάρχουσα με βάση το κέρδος, τότε αυτή αποθηκεύεται και τα removeNumber και notImproved τίθενται ίσα με 1 και 0 αντίστοιχα, αλλιώς το notImproved αυξάνεται κατά 1. Στην συνέχεια, ακολουθεί το βήμα της διαταραχής που συνοδεύεται από κατάλληλη ενημέρωση των startNumber και removeNumber. Όταν πλέον ολοκληρωθεί ο δεύτερος βρόγχος όλες οι διαδρομές ξανά αρχικοποιούνται ώστε να περιέχουν μόνο την αρχή και το τέλος. Ο ψευδοκώδικας του αλγορίθμου δίνεται στον Αλγόριθμο 3.6.

Λόγω των περιορισμών των CR που εισάγει ο αλγόριθμος, οι παραγόμενες λύσεις αναμένεται να έχουν λιγότερο κέρδος από τους ILS και CSCRatio κυρίως σε στιγμιότυπα με μικρά χρονικά παράθυρα. Από την άλλη μεριά, λόγω των CR οι διαδρομές των λύσεων θα έχουν λίγες μετακινήσεις μεταξύ των clusters. Το γεγονός αυτό, είναι πολύ επιθυμητό σε ρεαλιστικά προβλήματα καθώς λιγότερες μετακινήσεις μεταξύ μακρινών POIs ισοδυναμούν με λιγότερες μετακινήσεις με μέσα μαζικής μεταφοράς και ταυτόχρονα εξοικονόμηση χρημάτων. Ενώ, η διαδοχική επίσκεψη κόμβων του ίδιου cluster συνάγει μετάβαση με περπάτημα, κάτι που είναι πιο οικονομικό για έναν τουρίστα και ταυτόχρονα πιο οικολογικό. Εκτός αυτού, ο αλγόριθμος γίνεται πιο γρήγορος, καθώς το βήμα εισαγωγής γίνεται πιο αποδοτικό. Αυτό συμβαίνει καθώς πλέον οι πιθανές θέσεις εισαγωγής ενός κόμβου στην λύση είναι λιγότερες από ό,τι στον ILS και στον CSCRatio.

Αλγόριθμος 3.6 Ο αλγόριθμος **CSCRoutes**(numberOfClusters,maxIterations)

Προπαρασκευαστικό Στάδιο
εκτέλεσε τον αλγόριθμο global k-means με παράμετρο numberOfClusters
Δημιούργησε την λίστα listOfClusterSets

Online Διαδικασία
bestSolution $\leftarrow$ null
bestProfit $\leftarrow 0$
Αρχικοποίησε την currentSolution ως το σύνολο των K διαδρομών που αρχίζουν στον κόμβο u_1 και τερματίζουν στον κόμβο u_N.
Ενόσω η listOfClusterSets δεν είναι άδεια **Κάνε**
  ClusterSet $\leftarrow$ listOfClusterSets.pop
  RouteInitPhase(ClusterSet)
  notImproved $\leftarrow 0$
  startNumber $\leftarrow 1$
  removeNumber $\leftarrow 1$
  Ενόσω notImproved $<$ maxIterations **Κάνε**
    Ενόσω η currentSolution δεν είναι τοπικά βέλτιστη **Κάνε**
      CSCRoutes_Insert
    Τέλος Ενόσω
    currentProfit $\leftarrow$ το κέρδος της currentSolution
    Αν currentProfit $>$ bestProfit **Τότε**
      bestSolution $\leftarrow$ currentSolution
      bestProfit $\leftarrow$ currentProfit
      removeNumber $\leftarrow 1$
      notImproved $\leftarrow 0$
    Αλλιώς
      notImproved $\leftarrow$ notImproved $+ 1$
    Τέλος Αν
    Αν removeNumber $> \frac{\text{μέγεθος της μέγιστης διαδρομής}}{2}$ **Τότε**
      removeNumber $\leftarrow 1$
    Τέλος Αν
    Shake(removeNumber,startNumber)
    startNumber $\leftarrow$ startNumber $+$ removeNumber
    removeNumber $\leftarrow$ removeNumber $+ 1$
    smallestSize $\leftarrow$ το μέγεθος της μικρότερης διαδρομής στην λύση
    Αν startNumber $\geq$ smallestSize **Τότε**
      startNumber $\leftarrow$ startNumber $-$ smallestSize
    Τέλος Αν
  Τέλος Ενόσω
  Αφαίρεσε όλους τους κόμβους της currentSolution εκτός από την αρχή και το τέλος κάθε διαδρομής
Τέλος Ενόσω
Επέστρεψε bestSolution

3.5 Πειραματικά αποτελέσματα

3.5.1 Στιγμιότυπα

Για την αξιολόγηση των αλγορίθμων έχουν χρησιμοποιηθεί δημοσιευμένα στιγμιότυπα [62] όπως και καινούργια στιγμιότυπα που παράχθηκαν για να προσομοιωθούν πραγματικές τουριστικές τοπολογίες. Με βάση τα στιγμιότυπα του Solomon [73] (c101-c109, r101-r112, rc101-rc108) για το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα (vehicle routing problem with time windows) [52] και τα στιγμιότυπα των Cordeau et al. [22] (pr01-pr10), οι Righini και Salani [71] σχεδίασαν στιγμιότυπα για το OPTW, την ειδική περίπτωση του TOPTW με μία διαδρομή. Οι Montemanni και Gambardella [62] δημιούργησαν επιπλέον τα στιγμιότυπα (c201-c208, r201-r211, rc201-rc208) βασιζόμενοι στα στιγμιότυπα του Solomon και τα στιγμιότυπα (pr11-pr20) βασιζόμενοι στα στιγμιότυπα των Cordeau et al.

Όλα τα στιγμιότυπα του Solomon και οι επεκτάσεις τους (c*,r*,rc*) περιέχουν 100 κόμβους, ενώ το πλήθος των κόμβων στα στιγμιότυπα των Cordeau et al. κυμαίνονται από 48-288. Το μέγιστο επιτρεπόμενο μήκος των διαδρομών είναι 1236, 230, 240 για τα c1*, r1*, rc1*, 3390, 1000 και 960 για τα c2*, r2*, rc2* και 1000 για τα pr*. Επιπλέον, το μέσο μήκος των χρονικών παραθύρων των κόμβων είναι 321, 87, 85 για τα c1*, r1*, rc1*, 921, 454, 370 για τα c2*, r2*, rc2* και 135, 269 για τα pr0*, pr1*. Το πλήθος των διαδρομών σε όλα τα στιγμιότυπα κυμαίνεται από 1-4. Τα στιγμιότυπα μπορούν να βρεθούν στο `http://www.mech.kuleuven.be/en/cib/op/`.

Τα παραπάνω στιγμιότυπα μπορούν να χρησιμοποιηθούν για την αξιολόγηση αλγορίθμων TOPTW από θεωρητική σκοπιά. Επιπλέον, το πλήθος των κόμβων που περιέχουν είναι κατάλληλο για να αναπαρασταθεί ένα σύνολο αξιοθέατων μιας πόλης. Παρ' όλα αυτά δεν μπορούν να μοντελοποιήσουν ρεαλιστικά σενάρια τουριστικών προορισμών. Αυτό συμβαίνει γιατί τα χρονικά παράθυρα επίσκεψης στους κόμβους είναι συνήθως πολύ μικρά (π.χ. 85 για τα στιγμιότυπα rc1*) και παραμένουν τα ίδια για κάθε διαδρομή (ενώ π.χ. ένα μουσείο μπορεί να είναι κλειστό μια μέρα της βδομάδα ενώ όλες τις υπόλοιπες να είναι ανοιχτό 09:00-14:00). Ακόμα, το κέρδος των κόμβων δεν εξαρτάται από τον χρόνο παραμονής, ενώ το επιτρεπόμενο μήκος της διαδρομής είναι συνήθως πολύ μεγάλο (π.χ. είναι 3390 για τα στιγμιότυπα c2*).

Με σκοπό να μοντελοποιηθούν νέα ρεαλιστικά προβλήματα τουριστικών διαδρομών, δημιουργήθηκαν καινούργια στιγμιότυπα (t101-t150, t201-t250) με πλήθος κόμβων που κυμαίνονται από 100-200. Η τοποθέτηση των κόμβων στον χώρο προσομοιώνει πραγματικές τοπολογίες, όπου συνηθίζεται αρκετά αξιοθέατα να είναι σε κοντινές αποστάσεις μεταξύ τους. Για τον λόγο αυτό θεωρούνται 1-10 εικονικά κέντρα (δεν αποτελούν κόμβους του γράφου) και το 80% των κόμβων τοποθετούνται σε κοντινή απόσταση από κάποιο τυχαία επιλεγμένο εικονικό κέντρο. Οι υπόλοιποι κόμβοι τοποθετούνται τυχαία στο επίπεδο. Η χρονική διάρκεια της επίσκεψης σε κάθε κόμβο τίθεται από 1-120 λεπτά, ενώ το κέρδος κάθε κόμβου παίρνει μια τιμή από το διάστημα [1,100] και είναι ανάλογο της χρονικής διάρκειας της επίσκεψης σε αυτό. Τα στιγμιότυπα έχουν ρεαλιστικά χρονικά παράθυρα για κάθε μία από τις μέρες τις εβδομάδας, το 50% των κόμβων είναι ανοιχτό όλη την ημέρα (π.χ. πάρκα), ενώ ένα σύνολο αξιοθεάτων είναι κλειστό τα σαββατοκύριακα ή μια μέρα την βδομάδα. Τα αξιοθέατα που δεν είναι ανοιχτά κάθε μέρα, έχουν χρονικό παράθυρο επίσκεψης 08:30-17:00 για όλες τις μέρες που είναι ανοιχτά. Το επιτρεπόμενο χρονικό περιθώριο των διαδρομών είναι 10 ώρες για τα t1* στιγμιότυπα και 5 ώρες για τα t2*. Ενώ, το πλήθος των διαδρομών κυμαίνεται από 1–3. Τα στιγμιότυπα μπορούν να βρεθούν στο `http://www2.aegean.gr/dgavalas/public/op_instances/`.

3.5.2 Αποτελέσματα

Οι αλγόριθμοι που υλοποιήθηκαν είναι οι ILS, CSCRatio και CSCRoutes. Η υλοποίηση τους έγινε στην γλώσσα προγραμματισμού C++. Η εκτέλεση των αλγορίθμων έγινε στα στιγμιότυπα που περιγράφτηκαν στην προηγούμενη ενότητα χρησιμοποιώντας έναν προσωπικό υπολογιστή Intel Core i5 με 2,5 GHz επεξεργαστή και 4 GB RAM. Οι αλγόριθμοι αξιολογούνται στα εξής:

i) Στο κέρδος της λύσης που παράγουν.

ii) Στο πλήθος μετακινήσεων μεταξύ clusters στις διαδρομές τους, δηλαδή μετακινήσεις μεταξύ κόμβων διαφορετικών clusters.

iii) Στον χρόνο εκτέλεσης.

Ο κυρίαρχος στόχος των αλγορίθμων είναι η μεγιστοποίηση του κέρδους (i). Το πλήθος των μετακινήσεων (ii) έχει επιλεγεί σαν κριτήριο καθώς η μετακίνηση από έναν κόμβο ενός cluster σε κόμβο ενός διαφορετικού cluster μπορεί να μοντελοποιήσει μια μακρινή μεταφορά μέσα στην πόλη που να συνοδεύεται από χρήση μέσων μαζικής μεταφοράς, οπότε η ελαχιστοποίηση του πλήθους των μετακινήσεων είναι ένας σημαντικός παράγοντας στις τουριστικές διαδρομές. Τέλος, οι αλγόριθμοι για να είναι εφαρμόσιμοι σε εφαρμογές διαδικτύου, είναι σημαντικό να καταναλώνουν τον λιγότερο δυνατό χρόνο εκτέλεσης (iii).

Για τους αλγορίθμους CSCRatio και CSCRoutes έχουν χρησιμοποιηθεί οι ποσότητες numberOfClusters $= \frac{N}{10}$ έτσι ώστε κάθε cluster να έχει κατά μέσο όρο 10 κόμβους. Στην λίστα listOfClusterSets έχουν εισαχθεί $\lceil \text{numberOfClusters}/K \rceil$ μη επικαλυπτόμενες $K-$άδες από clusters που επιλέγονται τυχαία. Ενώ, το μέγιστο πλήθος επαναλήψεων χωρίς βελτίωση της λύσης έχει τεθεί ως maxIterations $= \frac{400}{|\text{listOfClusterSets}|} \cdot \frac{K+1}{2 \cdot K}$. Το πλήθος των επαναλήψεων τίθεται αντιστρόφως ανάλογο του πλήθους των $K-$άδων στην λίστα listOfClusterSets καθώς τουλάχιστον αυτό το πλήθος των επαναλήψεων εκτελείται ο δεύτερος βρόγχος για κάθε μία από τις $K-$άδες στην λίστα. Οπότε, οι συνολικές επαναλήψεις της τοπικής αναζήτησης συνοδευόμενης από το βήμα διαταραχής είναι τουλάχιστον $400 \cdot \frac{K+1}{2 \cdot K}$. Επίσης, οι επαναλήψεις φθίνουν όσο αυξάνεται το πλήθος των διαδρομών χρησιμοποιώντας τον παράγοντα $\frac{K+1}{2 \cdot K}$ καθώς σε στιγμιότυπα με μεγαλύτερο συνολικό διαθέσιμο χρόνο σε όλες τις διαδρομές οι βέλτιστες λύσεις μπορού να βρεθούν με μικρό αριθμό επαναλήψεων.

Στο παράρτημα Α′ παρουσιάζονται τα αναλυτικά αποτελέσματα των αλγορίθμων ILS, CSCRatio και CSCRoutes. Τα αποτελέσματα για τα στιγμιότυπα του Solomon δίνονται στους πίνακες Α′.1 – Α′.4 για 1 – 4 διαδρομές αντίστοιχα. Τα αποτελέσματα για τα στιγμιότυπα των Cordeau et al. δίνονται στους πίνακες Α′.5 – Α′.8 για 1 – 4 διαδρομές. Σε κάθε έναν από τους παραπάνω πίνακες οι δύο πρώτες στήλες δείχνουν το στιγμιότυπο στο οποίο εκτελέστηκαν οι αλγόριθμοι και το πλήθος των clusters που παράχθηκαν από τον global k-means. Μετά ακολουθούν τρεις τριάδες στηλών για τους ILS, CSCRatio και CSCRoutes αντίστοιχα. Σε κάθε τριάδα δίνονται με την σειρά το κέρδος που είχε ο αλγόριθμος για το συγκεκριμένο στιγμιότυπο, το πλήθος των μετακινήσεων που παράχθηκαν και ο χρόνος εκτέλεσης του αλγορίθμου σε ms. Τα αναλυτικά αποτελέσματα για τα νέα στιγμιότυπα δίνονται στου πίνακες Α′.9 και Α′.10. Τα δεδομένα των πινάκων έχουν παρόμοια δομή με τους πίνακες Α′.1 – Α′.8 με την προσθήκη μιας επιπλέον στήλης, πριν τις τριάδες των αποτελεσμάτων των αλγορίθμων, που δείχνει το πλήθος των διαδρομών που αναζητούνται.

Στον πίνακα 3.1 παρουσιάζεται η μέση ποσοστιαία απόκλιση του CSCRatio από τον ILS για κάθε ένα από τα σύνολα των στιγμιοτύπων, δηλαδή το c1* αντιπροσωπεύει το σύνολο των στιγμιοτύπων c101-c109, το c2* τα c201-c208 κτλπ. Η ποσοστιαία απόκλιση στο κέρδος των δύο αλγορίθμων ορίζεται ως $100 \cdot \frac{\text{κέρδος CSCRatio} - \text{κέρδος ILS}}{\text{κέρδος ILS}}$, στις μετακινήσεις ως $100 \cdot \frac{\text{μετακινήσεις ILS} - \text{μετακινήσεις CSCRatio}}{\text{μετακινήσεις ILS}}$ και στον χρόνο εκτέλεσης ως $100 \cdot \frac{\text{χρόνος ILS} - \text{χρόνος CSCRatio}}{\text{χρόνος ILS}}$. Από τους παραπάνω ορισμούς προκύπτει ότι η απόκλιση στο κέρδος γίνεται θετική όταν ο CSCRatio δίνει πιο κερδοφόρα λύση, ενώ η απόκλιση στις μετακινήσεις και τους χρόνους γίνεται θετική όταν ο CSCRatio παράγει διαδρομές με λιγότερες μετακινήσεις και εκτελείται σε συντομότερο χρόνο αντίστοιχα. Ο CSCRatio παράγει λύσεις με περισσότερο κέρδος από ότι ο ILS. Αυτό φαίνεται ξεκάθαρα σε στιγμιότυπα με λίγες διαδρομές και μικρό επιτρεπόμενο χρονικό περιθώριο, π.χ. το rc1* για 1 διαδρομή όπου ο CSCRatio υπερτερεί του ILS κατά 2.04%. Το ποσοστό αυτό είναι πολύ υψηλό αν αναλογιστεί κανείς ότι ο ILS

έχει μέση απόκλιση από τις μέγιστες υπολογισμένες τιμές για τα στιγμιότυπα των Solomon και Cordeau et al. λιγότερο από 2% [80]. Επίσης, στα καινούργια στιγμιότυπα ο CSCRatio υπερισχύει του ILS, π.χ. στο t2* έχει απόκλιση 2%. Τα αποτελέσματα αυτά οφείλονται κυρίως στο ότι ο ILS δεν μπορεί να εξερευνήσει καλά το σύνολο των λύσεων. Είτε παγιδεύεται σε μακρινούς κόμβους με μεγάλο κέρδος είτε δεν μπορεί να εξερευνήσει μακρινές γειτονιές με καλή συσσώρευση κόμβων με μεγάλο κέρδος. Την ίδια στιγμή, ο CSCRatio χρησιμοποιώντας την λίστα listOfClusterSets και το βήμα RouteInitPhase υπερκαλύπτει αυτές τις δυσκολίες. Επιπλέον, η εναλλαγή των τιμών της clusterParameter δίνει επίσης την δυνατότητα της εξερεύνησης του συνόλου των λύσεων σε μεγαλύτερο βάθος. Εκτός από το κέρδος, ο CSCRatio παράγει λύσεις με λιγότερες μετακινήσεις μεταξύ των clusters στην πλειονότητα των περιπτώσεων κυρίως λόγω της εύνοιας που προσφέρει στην εισαγωγή κόμβων πριν και μετά από κόμβους του ίδιου cluster. Αυτό γίνεται πολύ φανερό σε στιγμιότυπα με πολλές διαδρομές και πολύ διαθέσιμο χρόνο όπως τα c2*, r2* με 4 διαδρομές όπου και οι δύο αλγόριθμοι βρίσκουν το βέλτιστο αλλά ο CSCRatio εισάγει τους κόμβους από το ίδιο cluster διαδοχικά. Όσον αφορά τον χρόνο εκτέλεσης των αλγορίθμων, παρατηρούνται παρόμοιοι χρόνοι εκτέλεσης με εξαίρεση τα στιγμιότυπα με πολύ μεγάλο χρονικό περιθώριο και πολλές διαδρομές όπου ο ILS υπερτερεί του CSCRatio. Στα στιγμιότυπα αυτά η βέλτιστη λύση βρίσκεται και από τους δύο άμεσα, για παράδειγμα στα στιγμιότυπα r2* και rc2* με 4 επαναλήψεις και οι δύο αλγόριθμοι έχουν αρκετό διαθέσιμο χρόνο για να εισάγουν όλους τους κόμβους στην διαδρομή και αυτό γίνεται στην πρώτη επανάληψη. Ο CSCRatio τότε εκτελεί περισσότερες επαναλήψεις (250) από τον ILS (150), σε κάθε μία από τις οποίες στο βήμα διαταραχής του CSCRatio απομακρύνονται περισσότεροι κόμβοι από την λύση από ότι στον ILS. Αυτό συμβαίνει καθώς σε αυτά τα στιγμιότυπα το κλάσμα $\frac{N}{3k}$ γίνεται μικρότερο από το μισό του μεγέθους της μέγιστης διαδρομής μιας λύσης του CSCRatio. Ακολουθώντας την ίδια λογική, η τοπική αναζήτηση χρειάζεται περισσότερες επαναλήψεις για τον CSCRatio μέχρι να βρει τοπικό ελάχιστο οπότε καταναλώνει περισσότερο χρόνο.

Πίνακας 3.1: Μέση ποσοστιαία απόκλιση του CSCRatio σε σύγκριση με τον ILS

	Κέρδος(%)				Μετακινήσεις (%)				Χρόνος(%)			
Name	1	2	3	4	1	2	3	4	1	2	3	4
c1*	0.21	0.32	0.53	0.68	-0.2	-0.01	3.12	6.21	-40.8	8.45	35	24.9
c2*	0.84	0.79	0.29	0	19.1	12.6	12.3	20	-4.29	-18.2	-101	-398
r1*	0.79	0.91	-0.57	0.33	4.96	4.55	-0.11	2.71	-20.7	18.1	39.3	21.9
r2*	0.11	0.47	0.03	0	9.78	9.86	10.8	14	-4.88	-120	-305	-608
rc1*	2.04	0.87	0.81	-0.47	9.17	3.75	4.81	4.94	-1.07	36.3	34.7	44.7
rc2*	0.45	-0.34	0.32	0	5.49	1.83	0.48	5.49	11.9	-38.4	-197	-416
pr*	1.46	-0.02	0.4	0.9	-0.72	-9.99	4.5	4.62	29.1	27.4	7.44	-27.4
t1*	0.28				2.19				-5.27			
t2*	2				-13.2				8.33			

Η μέση ποσοστιαία απόκλιση του CSCRoutes, σε σύγκριση με τον ILS παρουσιάζεται στον Πίνακα 3.2. Η μέση απόκλιση στο κέρδος, τις μετακινήσεις και στον χρόνο εκτέλεσης ορίζεται όπως και στην σύγκριση του CSCRatio με τον ILS. Ο ILS υπερτερεί ξεκάθαρα του CSCRoutes σε κέρδος όπως ήταν αναμενόμενο έχοντας υπ' όψιν ότι ο CSCRoutes περιορίζει τις πιθανές του λύσεις λόγω των cluster routes που δημιουργεί σε κάθε διαδρομή. Τα στιγμιότυπα όπου ο ILS υπερτερεί περισσότερο σε κέρδος είναι εκείνα που συνδυάζουν μεγάλο επιτρεπόμενο μήκος διαδρομών (σε σχέση με τα χρονικά κόστη μεταξύ των κόμβων) και ταυτόχρονα κόμβους με μικρά χρονικά παράθυρα, όπως π.χ. τα στιγμιότυπα r2* με 1 διαδρομή όπου οι παραγόμενες λύσεις από τον ILS είναι κατά 15.5% πιο κερδοφόρες από του CSCRoutes. Σε αυτά τα στιγμιότυπα ο CSCRoutes δεν μπορεί να επισκεφτεί διαδοχικά πολλούς κόμβους από το ίδιο cluster,

λόγω των χρονικών παραθύρων των κόμβων, παράγοντας με αυτόν τον τρόπο λύσεις με μικρότερο κέρδος από ότι ο ILS. Αντίθετα, όταν υπάρχει πολύς χρόνος για επίσκεψη και ταυτόχρονα μεγάλα χρονικά παράθυρα στους κόμβους ο αλγόριθμος δεν υπολείπεται τόσο του ILS, π.χ. στα στιγμιότυπα r2* με 4 διαδρομές η ποσοστιαία απόκλιση στο κέρδος έχει γίνει 1.21%. Αυτό γίνεται επίσης πολύ εμφανές στα καινούργια στιγμιότυπα που δημιουργήθηκαν. Στα t1* ο CSCRoutes υπολείπεται του ILS μόλις 0.52%, ενώ το πιο αξιοσημείωτο αποτέλεσμα όσον αφορά το κέρδος είναι ότι ο CSCRoutes υπερτερεί κατά 1.91% του ILS στα στιγμιότυπα t2* που έχουν μεγάλα χρονικά παράθυρα σε σχέση με τον διαθέσιμο χρόνο στις διαδρομές. Σε αυτά τα στιγμιότυπα τα χρονικά παράθυρα δεν αποτρέπουν τον CSCRoutes να φτιάξει μεγάλες CR και σε συνδυασμό με το ότι ο αλγόριθμος επισκέπτεται όλα τα clusters (λόγω του βήματος RouteInitPhase), παράγονται λύσεις με μεγαλύτερο κέρδος από τον ILS. Επιπλέον, ο αλγόριθμος επιτυγχάνει τους στόχους βάση των οποίων σχεδιάστηκε, που είναι η ελαχιστοποίηση των μετακινήσεων μεταξύ των clusters και η όσο το δυνατόν εξοικονόμηση χρόνου εκτέλεσης. Ο CSCRoutes έχει θετική απόκλιση σε όλα τα σύνολα των στιγμιοτύπων όσον αφορά στις μετακινήσεις εκτός από το t2* που εκεί κάποια μικρά στιγμιότυπα με πολύ λίγες μετακινήσεις επηρεάζουν αυτό το πρόσημο. Σε αυτά τα στιγμιότυπα πραγματοποιούνται περισσότερες μετακινήσεις μεταξύ των clusters λόγω του RouteInitPhase. Τελειώνοντας, ο χρόνος εκτέλεσης του CSCRoutes είναι σημαντικά καλύτερος από ότι του ILS στις περισσότερες περιπτώσεις εκτός κυρίως από τα c2*, r2* και rc2* για 4 διαδρομές για λόγους παρόμοιους με αυτούς που αναφέρθηκαν στην σύγκριση του CSCRatio με τον ILS.

Πίνακας 3.2: Μέση ποσοστιαία απόκλιση του CSCRoutes σε σύγκριση με τον ILS

	Κέρδος(%)				Μετακινήσεις (%)				Χρόνος(%)			
Name	1	2	3	4	1	2	3	4	1	2	3	4
c1*	-1.65	-3.59	-1.03	-1.36	19.2	22.1	23.8	20.2	-21.1	29	38.2	31.2
c2*	-0.82	0.79	0.14	0	36	30.5	25.7	37.7	65.5	57.9	10.4	-170
r1*	-1.2	-1.27	-2.37	-2.15	23	21.7	16.6	22.2	0.1	36.5	49.4	38.6
r2*	-15.5	-10.3	-3.79	-1.21	56.3	55.4	52.6	46.4	76.7	25.1	-77.7	-284
rc1*	1.06	-1.8	-1.26	-1.86	16.7	11.9	14.8	14.4	10.7	50.3	42.9	52.9
rc2*	-9.5	-12.5	-8.21	-2.63	39.7	42.6	44.5	45.2	76	51.1	-40	-203
pr*	-8.11	-8.11	-5.44	-4.8	35.5	34.1	32.6	32.4	62.2	62.9	42.9	14.6
t1*	-0.52				5.31				22.2			
t2*	1.91				-4.5				4.59			

Ο πίνακας 3.3 παρουσιάζει την μέση ποσοστιαία απόκλιση του CSCRoutes συγκρινόμενος με τον CSCRatio. Η απόκλιση στις τιμές του κέρδους, των μετακινήσεων και του χρόνου εκτέλεσης ακολουθεί την ίδια προσέγγιση με τους δύο προηγούμενους πίνακες. Θετική απόκλιση στο κέρδος σημαίνει πιο κερδοφόρες λύσεις για τον CSCRoutes, ενώ αντίθετα θετική απόκλιση στις μετακινήσεις και στον χρόνο εκτέλεσης σημαίνει λιγότερες μετακινήσεις και λιγότερος χρόνος εκτέλεσης, αντίστοιχα. Από τα αποτελέσματα προκύπτει ότι ο CSCRoutes υπολείπεται του CSCRatio ως προς το κέρδος ελάχιστα (λιγότερο από 1 %) στα καινούργια στιγμιότυπα, ενώ υπολείπεται αισθητά στα στιγμιότυπα των Solomon και Cordeau et al. Η αρνητική απόκλιση στο κέρδος συμβαίνει επειδή το σύνολο των λύσεων που παράγει ο CSCRoutes είναι μικρότερο του CSCRatio λόγω των περιορισμών που δημιουργούν τα CR. Αντίθετα, αυτός ο περιορισμός δημιουργεί διαδρομές με πολύ λιγότερες μετακινήσεις μεταξύ των clusters, για κάθε σύνολο στιγμιοτύπων, αφού κόμβοι του ίδιου cluster εμφανίζονται διαδοχικά στις διαδρομές. Τέλος, ο χρόνος εκτέλεσης του CSCRoutes είναι αισθητά καλύτερος του CSCRatio. Ο λόγος για αυτό, είναι ότι στο βήμα εισαγωγής του CSCRoutes ελέγχονται για την εισαγωγή ενός νέου κόμβου λιγότερες θέσεις, λόγω των περιορισμών των CR.

Πίνακας 3.3: Μέση ποσοστιαία απόκλιση του CSCRoutes σε σύγκριση με τον CSCRatio

	Κέρδος(%)				Μετακινήσεις (%)				Χρόνος(%)			
Name	1	2	3	4	1	2	3	4	1	2	3	4
c1*	-1.86	-3.9	-1.54	-2	17.5	22.3	20.7	14.9	13.4	21.8	4.79	7.04
c2*	-1.65	0.01	-0.14	0	20.5	20.4	15.8	21.7	66.4	63	54.7	45.7
r1*	-1.97	-2.14	-1.81	-2.47	16.5	17.6	16.8	19.5	15.9	22.2	16.5	19.1
r2*	-15.6	-10.7	-3.81	-1.21	51	50.5	46.4	36.5	77.9	66.6	56.6	45.6
rc1*	-0.96	-2.59	-2.04	-1.4	7.5	8.53	9.61	8.92	10.8	20.2	12.6	10.4
rc2*	-9.9	-12.2	-8.49	-2.63	34.8	41.5	44.1	40.9	72.3	64.4	53.5	41.5
pr*	-9.4	-8.04	-5.8	-5.64	31.9	38.2	28.6	29	47	48.8	40.2	35.8
t1*	-0.78				1.46				24.5			
t2*	-0.12				4.85				-5.25			

3.6 Συμπεράσματα

Σε αυτό το κεφάλαιο εισήχθηκαν δύο νέοι ευρετικοί αλγόριθμοι για το TOPTW, ο CSCRatio και ο CSCRoutes. Κύριος στόχος των αλγορίθμων είναι να παράγουν σχεδόν βέλτιστες λύσεις και να έχουν χρόνο εκτέλεσης αρκετά γρήγορο ώστε να μπορούν να χρησιμοποιηθούν σε εφαρμογές διαδικτύου. Οι αλγόριθμοι θεωρώντας ως πρότυπο τον ILS [80], που είναι ο πιο κατάλληλος αλγόριθμος για εφαρμογές διαδικτύου, αντιμετωπίζουν κάποιες από τις αδυναμίες του ομαδοποιώντας τους κόμβους με βάση την γεωγραφική τους θέση. Η επίσκεψη σε ομάδες κόμβων απομακρυσμένες από την αρχή και το τέλος των διαδρομών ευνοείται, εξερευνώντας με αυτόν τον τρόπο σε πιο πολύ βάθος το σύνολο των εφικτών λύσεων. Εκτός από την μεγιστοποίηση του κέρδους και την ελαχιστοποίηση του χρόνου εκτέλεσης ένας ακόμα σημαντικός παράγοντας είναι η ελαχιστοποίηση του πλήθους των μετακινήσεων μεταξύ clusters. Διαδρομές με λίγες μετακινήσεις μεταξύ μακρινών σημείων ενδιαφέροντος είναι επιθυμητές, καθώς σε ρεαλιστικά προβλήματα μια τέτοια μετακίνηση θα σήμαινε την χρήση μέσων μαζικής μεταφοράς. Για αυτόν τον λόγο, ένας από τους στόχους των αλγορίθμων είναι η δημιουργία υποδιαδρομών από κόμβους του ίδιου cluster, υποδιαδρομών που προσομοιώνουν μετακινήσεις με πεζοπορία.

Ο αλγόριθμος CSCRatio υπερισχύει των υπόλοιπων αλγορίθμων σε κέρδος. Εκτός από το κέρδος ο CSCRatio παρουσιάζει καλύτερα αποτελέσματα σε σχέση με τον ILS όσον αφορά τις μετακινήσεις μεταξύ των clusters ενώ ο χρόνος εκτέλεσης των δύο αλγορίθμων είναι παρόμοιος. Ο CSCRoutes από την άλλη μεριά υπολείπεται των άλλων δύο αλγορίθμων στο κέρδος. Όμως, είναι ο πιο γρήγορος από τους τρεις και παράγει διαδρομές με πολύ λιγότερες μετακινήσεις μεταξύ των clusters σε σχέση με τους υπόλοιπους αλγορίθμους.

Τα παραπάνω καταδεικνύουν ότι δεν υπάρχει κάποιος από τους αλγορίθμους που να υπερισχύει και των τριών. Η επιλογή του αλγορίθμου που θα χρησιμοποιηθεί θα βασιστεί στα κριτήρια της εφαρμογής που θα χρειαστεί να εκτελέσει. Αν η εφαρμογή έχει σαν στόχο να παράγει λύσεις με μεγάλο κέρδος επιτρέποντας ένα εύλογο χρόνο εκτέλεσης ο πιο ικανοποιητικός αλγόριθμος θα ήταν ο CSCRatio, ενώ αν η εφαρμογή απαιτεί περιορισμένο χρόνο εκτέλεσης και αναζητάει διαδρομές με διαδοχικούς κόμβους σε κοντινή απόσταση η καλύτερη επιλογή θα ήταν ο αλγόριθμος CSCRoutes.

Κεφάλαιο 4

Ευρετικοί αλγόριθμοι για το TDTOPTW

Το TDTOPTW [31] επεκτείνει το TOPTW [78] θεωρώντας χρονικά εξαρτημένα κόστη ακμών. Ο χρόνος μετάβασης από έναν αρχικό κόμβο σε έναν επόμενο εξαρτάται από την στιγμή αναχώρησης από τον αρχικό κόμβο. Το TDTOPTW αποτελεί κατάλληλη μοντελοποίηση του Προβλήματος του Σχεδιασμού Τουριστικών Διαδρομών σε μεγάλες αστικές πόλεις όπου η μετακίνηση μεταξύ των αξιοθεάτων γίνεται με χρήση μέσων μαζικής μεταφοράς. Η χρήση μέσων μαζικής μεταφοράς παράγει χρονικά κόστη μετακινήσεων διαφορετικά κατά την διάρκεια της ημέρας, π.χ. στις ώρες αιχμής το τρένο έχει πιο συχνά δρομολόγια από ότι τις υπόλοιπες ώρες της ημέρας.

Στο κεφάλαιο αυτό προτείνονται τρεις νέοι ευρετικοί αλγόριθμοι για το TDTOPTW, ο TDCSCRoutes, ο SlackCSCRoutes και ο AvgCSCRoutes. Οι αλγόριθμοι βασίζονται στον αλγόριθμο CSCRoutes (Ενότητα 3.4.2). Οι δύο πρώτοι τον επεκτείνουν ώστε να μπορεί να χειριστεί χρονικά εξαρτημένα κόστη ακμών ενώ ο τρίτος τον χρησιμοποιεί σαν υπορουτίνα. Οι στόχοι των αλγορίθμων είναι οι εξής:

- Η παραγωγή διαδρομών που δίνουν μεγάλη ευχαρίστηση στον τουρίστα.
- Η παραγωγή διαδρομών με λίγες μετακινήσεις με χρήση μέσων μαζικής μεταφοράς, ευνοώντας τις μετακινήσεις με περπάτημα, καθώς αυτό είναι οικονομικό για τους τουρίστες όπως επίσης και οικολογικό.
- Ο γρήγορος χρόνος εκτέλεσης, ώστε οι αλγόριθμοι να μπορούν να χρησιμοποιηθούν σε εφαρμογές διαδικτύου.

Η μόνη προϋπάρχουσα δουλειά για το TDTOPTW είναι η δουλειά των Garcia et al. [31], όπου και ορίζεται για πρώτη φορά το πρόβλημα. Οι συγγραφείς βασιζόμενοι στον ILS αλγόριθμο [80] για το TOPTW προτείνουν δύο διαφορετικές κατευθύνσεις για την επίλυση του TDTOPTW. Στην πρώτη κατεύθυνση ένα τυχαίο στιγμιότυπο του TDTOPTW ανάγεται σε ένα στιγμιότυπο του TOPTW αντικαθιστώντας τα χρονικά εξαρτημένα κόστη στις ακμές από το μέσο κόστος της ακμής. Ύστερα, μια λύση για το παραγμένο στιγμιότυπο του TOPTW δημιουργείται χρησιμοποιώντας τον ILS αλγόριθμο. Τέλος, η λύση αυτή μετατρέπεται σε μια εφικτή λύση για το TDTOPTW, εισάγοντας τα κατάλληλα κόστη στις ακμές των διαδρομών. Αν αυτό το επιδιορθωτικό βήμα οδηγήσει σε μη έγκυρη λύση, λόγω των χρονικών περιορισμών, τότε οι κόμβοι που δεν ικανοποιούν τα χρονικά τους παράθυρα απομακρύνονται από την λύση. Η δεύτερη προσέγγιση παίρνει υπ' όψιν τους πραγματικούς χρονικά εξαρτημένους χρόνους των ακμών, χρησιμοποιώντας την απλοποιητική παραδοχή ότι οι ώρες άφιξης των μέσων μαζικής

μεταφοράς στους σταθμούς είναι περιοδικές και ταυτόχρονα οι χρόνοι μετάβασης μεταξύ των σταθμών παραμένουν ίδιοι κατά την διάρκεια της ημέρας. Αυτές οι παραδοχές δεν μπορεί να ισχύουν σε μεγάλες πόλεις, όπου τα δρομολόγια των μέσων μαζικής μεταφοράς αλλάζουν κατά την διάρκεια της ημέρας, πυκνώνοντας σε ώρες αιχμής και αραιώνοντας τις απογευματινές ώρες, ενώ ταυτόχρονα οι χρόνοι μετακίνησης είναι πολύ μεγαλύτεροι σε ώρες αιχμής. Για αυτόν τον λόγο οι αλγόριθμοι που προτείνονται σε αυτό το κεφάλαιο συγκρίνονται μόνο με την πρώτη προσέγγιση των Garcia et al.

Το κεφάλαιο αυτό οργανώνεται ως εξής: Στην Ενότητα 4.1 δίνεται ο μαθηματικός ορισμός του προβλήματος. Στην Ενότητα 4.2 διατυπώνονται οι ευρετικοί αλγόριθμοι για το TDTOPTW. Στην Ενότητα 4.3 περιγράφονται τα πειραματικά αποτελέσματα των αλγορίθμων. Τέλος τα συμπεράσματα του κεφαλαίου παρατίθενται στην Ενότητα 4.4.

4.1 Μαθηματικός Ορισμός του TDTOPTW

Το TDTOPTW είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης. Τα δεδομένα του προβλήματος είναι τα εξής:

- Ένας θετικός ακέραιος K που δείχνει το πλήθος των διαδρομών που πρέπει να κατασκευαστούν.
- Ένας πλήρης κατευθυνόμενος γράφος $G = (V, A)$ όπου $V = \{u_1, u_2, \ldots, u_N\}$ είναι το σύνολο των κορυφών του τέτοιος ώστε:
  - κάθε κορυφή u_i σχετίζεται με ένα κέρδος p_i και ένα χρόνο επίσκεψης v_i τέτοια ώστε η επίσκεψη στον κόμβο u_i συνοδεύεται από συλλογή κέρδους p_i και προϋποθέτει την παραμονή στον κόμβο για χρόνο v_i.
  - κάθε u_i έχει ένα χρονικό παράθυρο για κάθε μέρα $[R_{im}, D_{im}], m = 1, 2, \ldots, K$ με $R_{im} \leq D_{im}$ μέσα στο οποίο μπορεί να ξεκινήσει η επίσκεψη στον κόμβο.
  - Κάθε κορυφή u_i σχετίζεται με μια γεωγραφική τοποθεσία, έχοντας συντεταγμένες $[x_i, y_i]$.
- Κάθε διαδρομή $r_i, i = 1, 2, \ldots, K$ έχει ως αφετηρία τον κόμβο $s_i \in V$ και τερματισμό τον κόμβο $t_i \in V$ όπως επίσης και ένα μέγιστο επιτρεπόμενο μήκος $B_i, i = 1, \ldots, K$. Οι αρχικοί και τελικοί κόμβοι διαφορετικών διαδρομών δεν είναι αναγκαίο να ταυτίζονται.

Επιπλέον, για κάθε ζεύγος κόμβων $(u_i, u_j) \in A$ είναι δοσμένα ο χρόνος πεζοπορίας από το u_i στο u_j walking$_{ij}$ (σε περίπτωση που είναι πολύ μεγάλος θεωρείται ∞) όπως επίσης και ένα σύνολο S_{ij} μεγιστικών ζευγαριών (αναχώρηση, κόστος) $(\text{dep}_l^{ij}, \text{trav}_l^{ij}), l = 1, 2, \ldots, |S_{ij}|$ σε αύξουσα σειρά ως προς την ώρα της αναχώρησης. Τα ζευγάρια (αναχώρηση, κόστος) αναπαριστούν την ώρα αναχώρησης και το χρονικό κόστος μετακίνησης από τον u_i στον u_j χρησιμοποιώντας μέσα μαζικής μεταφοράς. Ένα ζεύγος a=(dep, trav) θεωρείται μεγιστικό αν είναι έγκυρο, δηλαδή αν το trav αναπαριστά το ελάχιστος κόστος μετάβασης από τον u_i στον u_j ξεκινώντας από τον κόμβο u_i την χρονική στιγμή dep και επίσης αν δεν υπάρχει επόμενη χρονική στιγμή αναχώρησης που έχει ώρα άφιξης στον κόμβο u_j μικρότερη ή ίση με την άφιξη του a, δηλαδή αν δεν υπάρχει (dep‘, trav‘) με dep‘ > dep και dep‘ +trav‘ $\leq$ dep +trav. Από τον ορισμό του, το σύνολο S_{ij} έχει την fifo (first in first out) ιδιότητα καθώς για κάθε δύο ζεύγη $(\text{dep}_l^{ij}, \text{trav}_l^{ij})$ και $(\text{dep}_{l+m}^{ij}, \text{trav}_{l+m}^{ij})$ η αναχώρηση από τον κόμβο u_i για τον κόμβο u_j

την χρονική στιγμή dep_l^{ij} θα οδηγήσει σε νωρίτερη άφιξη από ότι αν η αναχώρηση γινόταν την χρονική στιγμή dep_{l+m}^{ij}.

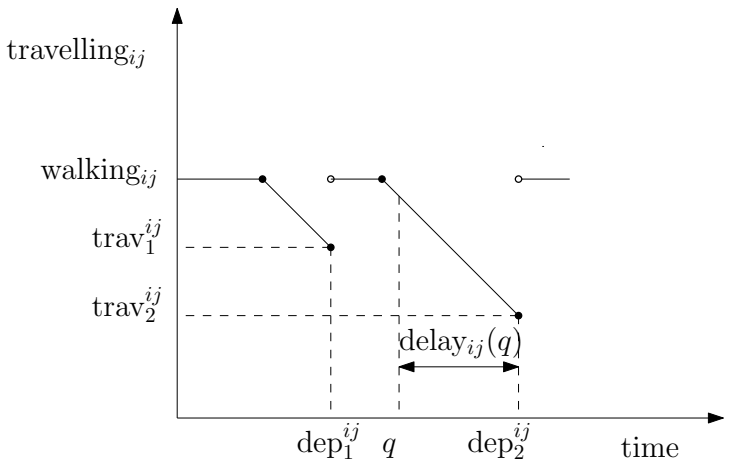
Για τυχαία χρονική στιγμή q ο βέλτιστος χρόνος αναχώρησης με μέσα μαζικής μεταφοράς είναι ο

$$\text{deptime}_{ij}(q) = \min_{l=1,2,\ldots,|S_{ij}|} \{\text{dep}_l^{ij} : q \leq \text{dep}_l^{ij}\}$$

Αντίστοιχα, ο χρόνος μετάβασης με μέσα μαζικής μεταφοράς την χρονική στιγμή q είναι ο $\text{travtime}_{ij}(q)$ τέτοιος ώστε $(\text{deptime}_{ij}(q), \text{travtime}_{ij}(q)) \in S_{ij}$. Ο χρόνος καθυστέρησης μέχρι την άφιξη της δημόσιας συγκοινωνίας είναι ο $\text{delay}_{ij}(q) = \text{deptime}_{ij}(q) - q$. Για τυχαία χρονική στιγμή αναχώρησης q από τον u_i στον u_j με $\text{dep}_l^{ij} < q \leq \text{dep}_{l+1}^{ij}$, ο συντομότερος χρόνος άφιξης στον u_j είναι είτε ο $q + \text{walking}_{ij}$, αν ο τουρίστας πάει με τα πόδια, είτε ο $\text{dep}_{l+1}^{ij} + \text{trav}_{l+1}^{ij}$ αν περιμένει μέχρι την χρονική στιγμή dep_{l+1}^{ij} και μετά συνεχίσει με την προβλεπόμενη διαδρομή με μέσα μαζικής μεταφοράς. Με την ίδια λογική ο συντομότερος χρόνος μετάβασης είναι είτε ο walking_{ij}, αν η μετάβαση γίνει με τα πόδια είτε ο $\text{delay}_{ij}(q) + \text{travtime}_{ij}(q)$, δηλαδή

$$\text{travelling}_{ij}(q) = \min\{\text{walking}_{ij}, \text{delay}_{ij}(q) + \text{travtime}_{ij}(q)\}$$

Ένα παράδειγμα της συνάρτησης του χρόνου μετάβασης από τον κόμβο u_i στον u_j δίνεται στην Εικόνα 4.1. Την χρονική στιγμή q ο βέλτιστος χρόνος αναχώρησης είναι $\text{deptime}_{ij}(q) = \text{dep}_2^{ij}$ ενώ το κόστος μετακίνησης είναι $\text{travelling}_{ij}(q) = \text{dep}_2^{ij} - q + \text{trav}_2^{ij}$.



Εικόνα 4.1: Παράδειγμα της συνάρτησης χρόνου μετάβασης από τον κόμβο u_i στον κόμβο u_j

Ο στόχος του TDTOPTW είναι να βρεθούν οι K διαδρομές $r_m, m = 1, \ldots, K$ που να ξεκινούν από το s_m, να καταλήγουν στο t_m, να έχουν μήκος το πολύ B_m και να συλλέγουν το μέγιστο δυνατό κέρδος. Οι διαδρομές πρέπει να είναι μη επικαλυπτόμενες, εκτός από πιθανά ίδια άκρα, και να επισκέπτονται τους κόμβους τους κατά την διάρκεια των χρονικών τους παραθύρων.

4.2 Προτεινόμενοι αλγόριθμοι

Για να μπορεί να εκτελεστεί ένα αποδοτικό βήμα εισαγωγής σε κάθε κόμβο u_i που ανήκει σε κάποια διαδρομή αποθηκεύονται οι εξής πληροφορίες:

- ο χρόνος άφιξης στον κόμβο, arrive_i
- ο χρόνος αναμονής στον κόμβο έως ότου να αρχίσει η επίσκεψη, wait_i
- ο χρόνος έναρξης της επίσκεψης, start_i
- ο χρόνος αναχώρησης από τον κόμβο, leave_i
- ο μέγιστος επιτρεπόμενος χρόνος έναρξης της επίσκεψης, maxStart_i, τέτοιος ώστε η έναρξη της επίσκεψης σε αυτόν τον κόμβο την συγκεκριμένη χρονική στιγμή να μην επηρεάζει την εγκυρότητα των χρονικών παραθύρων των επόμενων κόμβων, όπως επίσης και το επιτρεπόμενο χρονικό μήκος της διαδρομής.

Για κάθε διαδρομή $r_j, j = 1, 2, \ldots, K$ οι πληροφορίες των κόμβων στην διαδρομή ορίζονται αναδρομικά ως εξής: $\text{arrive}_{s_j} = 0$ και

- $\text{wait}_i = \max\{0, R_{ij} - \text{arrive}_i\}$
- $\text{start}_i = \text{arrive}_i + \text{wait}_i$
- $\text{leave}_i = \text{start}_i + v_i$
- $\text{arrive}_{\text{next}_i} = \text{leave}_i + \text{travelling}_{i,\text{next}_i}(\text{leave}_i)$

όπου με next_i συμβολίζεται ο δείκτης του κόμβου που έπεται του u_i στην διαδρομή.

Επίσης, $\text{maxStart}_{t_j} = B_j$ και

$$\text{maxStart}_i = \min\{D_{ij}, \max\{q : q + \text{travelling}_{i,\text{next}_i}(q) \leq \text{maxStart}_{\text{next}_i}\} - v_i\}$$

Η εισαγωγή του κόμβου u_i μετά τον κόμβο u_j στην διαδρομή r_m θα έχει κάποια χρονική επιβάρυνση κυρίως όσον αφορά το κομμάτι της διαδρομής από τον u_i στον επόμενό του, έστω u_k. Αυτή η χρονική επιβάρυνση θα είναι ίση με

$$\text{shift}_i^j = \text{travelling}_{ji}(\text{leave}_j) + \text{wait}_i^j + v_i + \text{travelling}_{ik}(\text{leave}_i^j) - \text{travelling}_{jk}(\text{leave}_j)$$

όπου wait_i^j και leave_i^j θα είναι οι χρόνοι αναμονής και αναχώρησης στον κόμβο u_i αν η εισαγωγή μετά τον κόμβο u_j πραγματοποιηθεί. Αυτή η κατανάλωση μπορεί να απορροφηθεί στους επόμενους κόμβους της διαδρομής είτε μέσω του χρόνου αναμονής των κόμβων λόγω των χρονικών τους παραθύρων είτε μέσω των delay χρόνων για την αναμονή δημόσιας συγκοινωνίας. Η εισαγωγή αυτή μπορεί να πραγματοποιηθεί αν και μόνο αν ικανοποιούνται οι επόμενες συνθήκες:

$$\text{leave}_j + \text{travelling}_{ji}(\text{leave}_j) \leq D_{im} \text{ και ταυτόχρονα } \text{shift}_i^j \leq \text{maxStart}_k - \text{arrive}_k$$

Ο ψευδοκώδικας για το shift του κόμβου u_i μετά τον κόμβο u_j στην διαδρομή r_m δίνεται στον Αλγόριθμο 4.1.

4.2.1 Ο αλγόριθμος TDCSCRoutes

Ο αλγόριθμος TDCSCRoutes επεκτείνει τον αλγόριθμο CSCRoutes, που αναφέρθηκε στο προηγούμενο κεφάλαιο. Το βήμα εισαγωγής του CSCRoutes τροποποιείται ώστε να μπορεί να χειριστεί χρονικά εξαρτημένα κόστη ακμών. Το βήμα διαταραχής παραμένει ίδιο ενώ η ανανέωση των μεταβλητών στους κόμβους μια λύσης γίνεται χρησιμοποιώντας τα χρονικά εξαρτημένα κόστη. Σε ένα προπαρασκευαστικό στάδιο γίνεται η ομαδοποίηση των κόμβων σε clusters με βάση την γεωμετρική τους θέση χρησιμοποιώντας τον global k-means αλγόριθμο [4, 57]. Επίσης, χρησιμοποιείται η δομή των

Αλγόριθμος 4.1 Ο υπολογισμός του shift_i^j στην διαδρομή r_m

shift $\leftarrow \infty$
tempArrive $\leftarrow \text{leave}_j + \text{travelling}_{ji}(\text{leave}_j)$
Αν tempArrive $\leq \text{D}_{im}$ **Τότε**
  tempWait $\leftarrow \max\{0, \text{R}_{im} - \text{tempArrive}\}$
  tempLeave $\leftarrow$ tempArrive + tempWait $+ v_i$
  $k \leftarrow \text{next}_j$
  tempShift $\leftarrow \text{travelling}_{ji}(\text{leave}_j) + \text{tempWait} + v_i + \text{travelling}_{ik}(\text{tempLeave}) - \text{travelling}_{jk}(\text{leave}_j)$
  Αν tempShift $\leq \text{maxStart}_k - \text{arrive}_k$ **Τότε**
    shift $\leftarrow$ tempShift
  Τέλος Αν
Τέλος Αν
Επέστρεψε shift

CR (cluster routes) που διατυπώθηκε στο προηγούμενο κεφάλαιο. Για λόγους πληρότητας αυτού του κεφαλαίου θα οριστούν ξανά και θα γενικευτούν κάποιες έννοιες που υπάρχουν στο προηγούμενο κεφάλαιο.

Ένα CR (cluster route) στην διαδρομή r που σχετίζεται με το cluster C (CR_C^r) είναι μια μέγιστική ακολουθία διαδοχικών κόμβων στην διαδρομή r από το cluster C. Ο TDCSCRoutes είναι σχεδιασμένος έτσι ώστε για κάθε cluster C και διαδρομή r της λύσης να δημιουργεί το πολύ ένα CR στην διαδρομή r που να σχετίζεται με το C. Η μόνη εξαίρεση σε αυτόν τον κανόνα είναι η περίπτωση που ο αρχικός s_r και ο τελικός t_r κόμβος της διαδρομής είναι στο ίδιο cluster, οπότε επιτρέπονται το πολύ δύο CR από αυτό το cluster, τα CR_f^r που έχει ως αφετηρία το s_r και το CR_l^r που έχει ως τερματισμό τον κόμβο t_r. Λόγω αυτού του κανόνα η εισαγωγή ενός κόμβου που δεν ανήκει στην λύση δεν μπορεί να πραγματοποιηθεί μετά από οποιονδήποτε επισκεπτόμενο κόμβο, αλλά μόνο στις θέσεις που η εισαγωγή του δεν θα παραβιάζει τα CR. Για τον λόγο αυτό σε κάθε βήμα εισαγωγής του αλγορίθμου είναι αναγκαίος ο άμεσος προσδιορισμός των εφικτών θέσεων εισαγωγής ενός κόμβου. Για να είναι αυτός ο προσδιορισμός αποδοτικός σε κάθε διαδρομή r θεωρείται μια λίστα clustersIn(r) με τα clusters που επισκέπτεται αυτή η διαδρομή, όπως επίσης και ο αρχικός και τελικός κόμβος κάθε CR. Επίσης, για κάθε κόμβο u_i θεωρείται ότι το cluster(u_i) αντιπροσωπεύει το cluster που ανήκει ο u_i. Τότε, οι πιθανές θέσεις εισαγωγής του κόμβου u_i στην διαδρομή r είναι οι εξής:

- cluster(s_r) = cluster(t_r)
  - Αν η clustersIn(r) περιέχει μόνο το cluster(s_r), τότε ο u_i μπορεί να εισαχθεί παντού στην r.
  - Αν η clustersIn(r) περιέχει τουλάχιστον δύο clusters και cluster(s_r) = cluster(u_i), τότε ο u_i μπορεί να εισαχθεί παντού στις CR_f^r και CR_l^r.
  - Αν η clustersIn(r) περιέχει τουλάχιστον δύο clusters, cluster(s_r) $\neq$ cluster(u_i) και cluster(u_i) $\notin$ clustersIn(r), τότε ο u_i μπορεί να εισαχθεί μετά από κάθε τελικό κόμβο ενός CR.
  - Αν η clustersIn(r) περιέχει τουλάχιστον δύο clusters, cluster(s_r) $\neq$ cluster(u_i) και cluster(u_i) $\in$ clustersIn(r), τότε ο u_i μπορεί να εισαχθεί μέσα στο CR που σχετίζεται με το cluster του.
- cluster(s_r) $\neq$ cluster(t_r)
  - Αν cluster(u_i) = cluster(s_r), τότε ο u_i μπορεί να εισαχθεί μέσα στην CR_f^r.
  - Αν cluster(u_i) = cluster(t_r), τότε ο u_i μπορεί να εισαχθεί μέσα στην CR_l^r.

- Αν cluster(u_i) $\neq$ cluster(s_r), cluster(u_i) $\neq$ cluster(t_r) και cluster(u_i) ανήκει στην λίστα clustersIn(r), τότε ο u_i μπορεί να εισαχθεί μέσα στην CR που σχετίζεται με το cluster(u_i).
- Αν το cluster(u_i) δεν ανήκει στην λίστα clustersIn(r), τότε ο u_i μπορεί να εισαχθεί μετά από κάθε τελικό κόμβο ενός CR.

Το κριτήριο για την εισαγωγή ενός κόμβου στο βήμα εισαγωγής του TDCSCRoutes (**TDCSCRoutes_Insert**) επεκτείνει το κριτήριο που χρησιμοποιεί ο CSCRoutes παίρνοντας υπ' όψιν επιπρόσθετες πληροφορίες για τα χρονικά εξαρτημένα κόστη στις ακμές και μια επιπλέον παράμετρο. Έστω η διαδρομή r_m και οι διαδοχικοί κόμβοι της u_j, u_k, u_l, οπότε $k = \text{next}_j, l = \text{next}_k$. Θα λέμε ότι η εισαγωγή του υποψήφιου για εισαγωγή κόμβου u_i μετά τον επισκεπτόμενο κόμβο u_j έχει βάρος weight_i^j που ισούται με

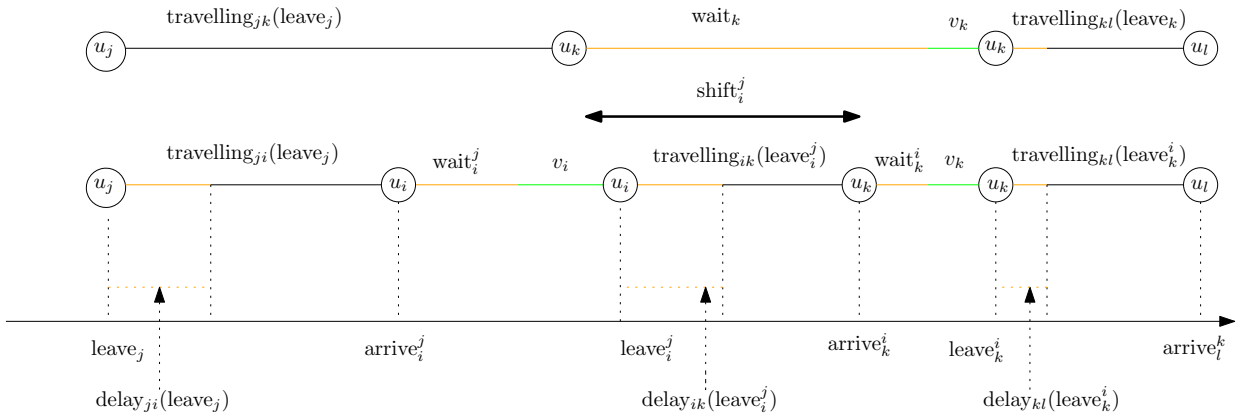
$$\text{weight}_i^j = \frac{p_i^2}{\text{shift}_i^j}[1 + \text{itPar}\frac{D_i^j + 1}{D_i^j + 2} + (1 - \text{itPar})\chi_{\leq}(\text{shift}_i^j, \text{wait}_k + \text{delay}_{kl}(\text{leave}_k))]$$

όπου η $\chi_{\leq}$ είναι η χαρακτηριστική συνάρτηση της ανισότητας $\leq$, δηλαδή $\chi_{\leq}(a, b) = 1$ αν $a \leq b$ και 0 αλλιώς. Επίσης, ορίζεται ως D_i^j ο αναξιοποίητος χρόνος που θα επιβαρυνθεί η διαδρομή με την εισαγωγή του u_i μετά τον κόμβο u_j, δηλαδή $D_i^j = \text{delay}_{ji}(\text{leave}_j) + \text{wait}_i^j + \text{delay}_{ik}(\text{leave}_i^j) + \text{wait}_k^i$. Τέλος, ορίζεται η παράμετρος itPar που ανάλογα με το πλήθος των επαναλήψεων παίρνει τιμές ώστε να ενισχύσει ή να αποδυναμώσει κάποιους όρους του κριτηρίου. Πιο συγκεκριμένα, αρχικά η παράμετρος τίθεται ίση με 1 ώστε να ευνοηθεί η εισαγωγή κόμβων με μεγάλο αναξιοποιήτο χρόνο. Ενώ η τιμή της μειώνεται κατά 0.5 κάθε $\frac{1}{3}$ επαναλήψεων χωρίς βελτίωση της λύσης, έτσι ώστε στο τέλος της διαδικασίας να ευνοείται η εισαγωγή κόμβων που καλύπτουν τα αναξιοποιήτα κενά. Η ιδέα αυτού του κριτηρίου εισαγωγής συνοψίζεται στα εξής: ο λόγος $\frac{p_i^2}{\text{shift}_i^j}$ χρησιμοποιείται ως παράγοντας καθώς αποτελεί ένα μέτρο του κέρδους που θα επιφέρει η εισαγωγή του κόμβου u_i έναντι του χρόνου που θα καταναλωθεί. Επιπλέον, ο λόγος $\frac{D_i^j+1}{D_i^j+2}$ χρησιμοποιείται ώστε όταν η itPar έχει μεγάλη τιμή να ευνοείται η εισαγωγή κόμβων με μεγάλο αναξιοποίητο χρόνο. Τέλος, χρησιμοποιείται ο όρος $\chi_{\leq}(\text{shift}_i^j, \text{wait}_k + \text{delay}_{kl}(\text{leave}_k))$ για να ευνοηθούν οι εισαγωγές σε θέσεις, όπου ο χρόνος κατανάλωσης απορροφάται από τον αναξιοποίητο χρόνο του επομένου, δηλαδή τον χρόνο αναμονής μέχρι να ξεκινήσει η επίσκεψη προστιθέμενο με τον χρόνο καθυστέρησης μέχρι την έλευση των μέσων μαζικής μεταφοράς. Στην Εικόνα 4.2 παρουσιάζονται τα διάφορα μεγέθη που χρησιμοποιούνται για τον υπολογισμό του βάρους εισαγωγής του u_i μετά τον κόμβο u_j.

Ο ψευδοκώδικας του βήματος εισαγωγής του TDCSCRoutes δίνεται στον Αλγόριθμο 4.2, ενώ ο ψευδοκώδικας όλου του αλγορίθμου δίνεται στον Αλγόριθμο 4.3.

4.2.2 Ο αλγόριθμος SlackCSCRoutes

Ο αλγόριθμος SlackCSCRoutes διαφοροποιείται από τον TDCSCRoutes στο κριτήριο που χρησιμοποιείται για το βήμα εισαγωγής. Το κριτήριο εισαγωγής του TDCSCRoutes είναι τοπικό, παίρνοντας υπ' όψιν κυρίως το κέρδος που θα συνεισφέρει η εισαγωγή ενός κόμβου στην λύση και την χρονική επιβάρυνση που θα έχει η διαδρομή μέχρι τον κόμβο που θα τον ακολουθήσει. Αντίθετα, στον SlackCSCRoutes το κριτήριο είναι σχεδιασμένο ώστε να είναι καθολικό, δηλαδή να παίρνει υπ' όψιν όλους όλους τους κόμβους μιας διαδρομής, για την εισαγωγή ενός νέου κόμβου.



Εικόνα 4.2: Παράδειγμα των χρονικά εξαρτημένων μεγεθών που εισέρχονται στην εισαγωγή του κόμβου u_i μετά τον u_j

Αλγόριθμος 4.2 Το βήμα εισαγωγής **TDCSCRoutes_Insert**(itPar)

Για κάθε μη εισηγμένο κόμβο u_i **Κάνε**
  Για κάθε διαδρομή r **Κάνε**
    Αν cluster(s_r) = cluster(t_r) **Τότε**
      Αν clustersIn(r) περιέχει μόνο το cluster(s_r) **Τότε**
        Ψάξε όλες τις θέσεις στην r για το μέγιστο βάρος
      Αλλιώς
        Αν cluster(s_r) = cluster(u_i) **Τότε**
          Ψάξε όλες τις θέσεις στις CR_f^r και CR_l^r για το μέγιστο βάρος
        Αλλιώς
          Αν cluster(u_i) $\notin$ clustersIn(r) **Τότε**
            Ψάξε όλες τις θέσεις που είναι τέλος CR για το μέγιστο βάρος
          Αλλιώς
            Ψάξε όλες τις θέσεις στην CR που σχετίζεται με το cluster(u_i) για το μέγιστο βάρος
          Τέλος Αν
        Τέλος Αν
      Τέλος Αν
    Αλλιώς ▷ cluster(s_r) $\neq$ cluster(t_r)
      Αν cluster(u_i) = cluster(s_r) **Τότε**
        Ψάξε όλες τις θέσεις στο CR_f^r για το μέγιστο βάρος
      Αλλιώς
        Αν cluster(u_i) = cluster(t_r) **Τότε**
          Ψάξε όλες τις θέσεις στο CR_l^r για το μέγιστο βάρος
        Αλλιώς
          Αν cluster(u_i) ανήκει στην λίστα clustersIn(r) **Τότε**
            Ψάξε όλες τις θέσεις στην CR που σχετίζεται με το cluster(u_i) για το μέγιστο βάρος
          Αλλιώς
            Ψάξε όλες τις θέσεις που είναι τελικοί κόμβοι κάποιας CR στην r για το μέγιστο βάρος
          Τέλος Αν
        Τέλος Αν
      Τέλος Αν
    Τέλος Αν
  Τέλος Για
Τέλος Για
Εισήγαγε τον κόμβο που παρουσίασε το μέγιστο βάρος στην αντίστοιχη θέση
Ενημέρωσε όλους τους χρόνους των κόμβων και όλες τις πληροφορίες για τα clusters στην διαδρομή που έγινε η εισαγωγή

Αλγόριθμος 4.3 Ο αλγόριθμος **TDCSCRoutes**(numberOfClusters,maxIterations)

Προπαρασκευαστικό Στάδιο
εκτέλεσε τον αλγόριθμο global k-means με παράμετρο numberOfClusters
Δημιούργησε την λίστα listOfClusterSets

Online Διαδικασία
bestSolution ← null
bestProfit ← 0
Κατασκεύασε τις αρχικές διαδρομές $r_m, m = 1, \ldots, K$, που περιέχουν μόνο τους s_m, t_m
Ενόσω η listOfClusterSets δεν είναι άδεια **Κάνε**
  ClusterSet ← listOfClusterSets.pop
  RouteInitPhase(ClusterSet)
  notImproved ← 0
  startNumber ← 1
  removeNumber ← 1
  Ενόσω notImproved $<$ maxIterations **Κάνε**
    itPar ← 1
    Αν notImproved $> \frac{\text{maxIterations}}{3}$ **Τότε**
      itPar ← itPar - 0.5
      Αν notImproved $> \frac{2\text{maxIterations}}{3}$ **Τότε**
        itPar ← itPar - 0.5
      Τέλος Αν
    Τέλος Αν
    Ενόσω η currentSolution δεν είναι τοπικά βέλτιστη **Κάνε**
      TDCSCRoutes_Insert(itPar)
    Τέλος Ενόσω
    currentProfit ← το κέρδος της currentSolution
    Αν currentProfit $>$ bestProfit **Τότε**
      bestSolution ← currentSolution
      bestProfit ← currentProfit
      removeNumber ← 1
      notImproved ← 0
    Αλλιώς
      notImproved ← notImproved $+ 1$
    Τέλος Αν
    Αν removeNumber $> \frac{\text{μεγέθους της μέγιστης διαδρομής}}{2}$ **Τότε**
      removeNumber ← 1
    Τέλος Αν
    Shake(removeNumber,startNumber)
    startNumber ← startNumber $+$ removeNumber
    removeNumber ← removeNumber $+ 1$
    smallestSize ← το μέγεθος της μικρότερης διαδρομής στην λύση
    Αν startNumber $\geq$ smallestSize **Τότε**
      startNumber ← startNumber $-$ smallestSize
    Τέλος Αν
  Τέλος Ενόσω
  Αφαίρεσε όλους τους κόμβους της currentSolution εκτός από την αρχή και το τέλος κάθε διαδρομής
Τέλος Ενόσω
Επέστρεψε bestSolution

Στον SlackCSCRoutes εισάγεται μια επιπλέον παράμετρος σε κάθε κόμβο που ανήκει στην λύση. Η παράμετρος σχετίζεται με το πόσος κενός χώρος υπάρχει για την εισαγωγή νέων κόμβων πριν από τον συγκεκριμένο. Πιο επίσημα, για κάθε κόμβο u_i που ανήκει σε κάποια διαδρομή θεωρείται η μεταβλητή

$$\text{slack}_i = \text{maxStart}_i - \text{arrive}_i$$

Όσο πιο μεγάλο είναι το slack_i τόσο πιο μεγάλη είναι η πιθανότητα να μπορέσει να εισαχθεί ένας κόμβος πριν από τον u_i, ενώ όταν αυτή η ποσότητα είναι κοντά στο 0, τότε η εισαγωγή νέων κόμβων πριν από τον u_i γίνεται σχεδόν αδύνατη.

Η εισαγωγή ενός νέου κόμβου u_i σε μια διαδρομή r επηρεάζει τις χρονικές μετα-

βλητές όλων των κόμβων της r. Ο χρόνος άφιξης στους κόμβους που θα έπονται του u_i μεγαλώνει καθώς προστίθεται στους χρόνους το κόστος ταξιδιού στον κόμβο u_i μαζί με τον χρόνο επίσκεψης στον κόμβο αυτό. Επιπλέον, ο χρόνος maxStart των κόμβων που θα προηγούνται του u_i μειώνεται καθώς εισάγεται ο επιπλέον περιορισμός στον χρόνο maxStart του u_i. Οπότε, η εισαγωγή ενός καινούργιου κόμβου σε μια διαδρομή έχει ως αποτέλεσμα να μειώνεται η μεταβλητή slack σε ένα υποσύνολο των κόμβων που ανήκουν σε αυτήν.

Έστω, ότι η διαδρομή r είναι η $(u_{n_1} = s_r, u_{n_2}, \dots u_{n_j}, \dots, u_{n_q} = t_r)$. Η εισαγωγή του u_i μετά το u_{n_j} θα έχει ως αποτέλεσμα, οι κόμβοι μετά το u_{n_j} να έχουν ένα καινούργιο arrive χρόνο, έστω τον χρόνο $\text{arrive}_{n_l}^{i,n_j}, l = j+1, \dots, q$, διατηρώντας το ίδιο maxStart, δηλαδή $\text{maxStart}_{n_l}^{i,n_j} = \text{maxStart}_{n_l}, l = j+1, \dots, q$. Επίσης, οι αρχικοί κόμβοι της διαδρομής εως τον κόμβο u_{n_j} διατηρούν το ίδιο arrive, δηλαδή $\text{arrive}_{n_l}^{i,n_j} = \text{arrive}_{n_l}, l = 1, \dots, j$ ενώ μειώνεται το maxStart τους, $\text{maxStart}_{n_l}^{i,n_j}, l = 1, \dots, j$. Τότε, το καινούργιο slack κάθε κόμβου της διαδρομής r μετά την εισαγωγή του κόμβου u_i μετά τον u_{n_j} θα είναι ίσο με $\text{slack}_{n_l}^{i,n_j} = \text{maxStart}_{n_l}^{i,n_j} - \text{arrive}_{n_l}^{i,n_j}, l = 1, 2, \dots, q$. Η ιδέα του αλγορίθμου είναι να επιλέγεται ως καλύτερη θέση εισαγωγής του κόμβου u_i η θέση στην οποία επιτυγχάνεται το μέγιστο μέσο slack από τους κόμβους της διαδρομής. Τότε, αν ο κόμβος u_i εισαχθεί μετά τον κόμβο u_{n_j} στην διαδρομή r, το μέσο slack των κόμβων στην διαδρομή θα είναι ίσο με $\text{avSlack}_i^{n_j} = \frac{\sum_{l=1}^{q} \text{slack}_{n_l}^{i,n_j} + \text{slack}_i^{n_j}}{q+1}$, όπου το $\text{slack}_i^{n_j}$ θα είναι το slack που θα έχει ο κόμβος u_i μετά την εισαγωγή του έπειτα από τον κόμβο u_{n_j}. Τότε, η θέση όπου το u_i επιτυγχάνει το μέγιστο avSlack θεωρείται η επιθυμητή θέση εισαγωγής του u_i και επιπλέον ορίζεται το $\text{avSlack}_i = \max_{n_j} \text{avSlack}_i^{n_j}$. Ως βέλτιστος κόμβος για εισαγωγή επιλέγεται ο κόμβος με την καλύτερη ποσότητα $\text{slackWeight}_i = p_i^2 \cdot \text{avSlack}_i$. Η ποσότητα αυτή επιλέγεται ώστε να εξασφαλιστεί η εισαγωγή ενός κόμβου με μεγάλο κέρδος που ταυτόχρονα να παρέχει την δυνατότητα εισαγωγής καινούργιων κόμβων στην διαδρομή.

Ο υπολογισμός του avgSlack για την εισαγωγή του κόμβου u_i έπειτα από τον κόμβο u_{n_j} προϋποθέτει τον υπολογισμό των χρόνων άφιξης, έναρξης της επίσκεψης και αναχώρησης από τον κόμβο u_i, όπως και της ενημέρωσης των χρόνων άφιξης, έναρξης και αναχώρησης από όλους τους κόμβους που θα έπονται του u_{n_j}. Εκτός από αυτό, πρέπει επίσης να υπολογιστεί η τιμή maxStart του u_i, όπως επίσης και οι νέες τιμές των maxStart των αρχικών κόμβων της διαδρομής έως και του u_{n_j}. Αυτοί οι υπολογισμοί κάνουν την χρήση του κριτηρίου αυτού μη αποδοτική και άρα ακατάλληλη για χρήση. Ας μην ξεχνάμε ότι στόχος των αλγορίθμων είναι η ελαχιστοποίηση του χρόνου εκτέλεσης καθώς οι αλγόριθμοι σχεδιάζονται για χρήση σε εφαρμογές διαδικτύου.

Για να γίνει το κριτήριο αποδοτικό χρησιμοποιείται μια προσέγγιση. Στην προσέγγιση δεν χρειάζεται ο επαναϋπολογισμός όλων των τιμών των μεταβλητών για όλους τους κόμβους, αλλά μόνο για τους γειτονικούς κόμβους της θέσης που θα γίνει η εισαγωγή. Ειδικότερα, για την εισαγωγή του κόμβου u_i έπειτα από τον κόμβο u_{n_j}, οπότε πριν από τον κόμβο $u_{n_{j+1}}$, θεωρείται ότι ο χρόνος άφιξης όλων των κόμβων $u_{n_l}, l = j+1, j+2, \dots, q$ θα αυξηθεί περίπου κατά $\text{shift}_i^{n_j}$, οπότε $\text{slack}_{n_l}^{i,n_j} \approx \text{slack}_{n_l} - \text{shift}_i^{n_j}, l = j+1, j+2, \dots, q$. Επίσης, θεωρείται ότι η τιμή του maxStart όλων των κόμβων πριν από τον u_{n_j} θα μειωθεί περίπου όσο θα μειωθεί η τιμή του u_{n_j}, δηλαδή $\text{maxStart}_{n_l}^{i,n_j} - \text{maxStart}_{n_l} \approx \text{maxStart}_{n_j}^{i,n_j} - \text{maxStart}_{n_j}, l = 1, 2, \dots, j-1$, οπότε $\text{slack}_{n_l}^{i,n_j} \approx \text{slack}_{n_l} + (\text{maxStart}_{n_j}^{i,n_j} - \text{maxStart}_{n_j}), l = 1, 2, \dots, j$, οπότε η προσέγγιση της ποσότητας

avSlack θα είναι πλέον ίση με

$$\text{appAvSlack}_i^{n_j} = \frac{\sum_{l=1}^{j}(\text{slack}_{n_l} + (\text{maxStart}_{n_j}^{i,n_j} - \text{maxStart}_{n_j})) + \text{slack}_i^{n_j} + \sum_{l=j+1}^{q}(\text{slack}_{n_l} - \text{shift}_i^{n_j})}{q+1}$$

όπου η ποσότητα αυτή είναι πλέον ίση με

$$\text{appAvSlack}_i^{n_j} = \frac{\sum_{l=1}^{q}\text{slack}_{n_l} + \text{slack}_i^{n_j} + j \cdot (\text{maxStart}_{n_j}^{i,n_j} - \text{maxStart}_{n_j}) - (q-j)\cdot \text{shift}_i^{n_j}}{q+1}$$

το τελευταίο κριτήριο είναι πλέον τοπικό, καθώς κρατώντας σε κάθε διαδρομή το άθροισμα των slacks, $\sum_{l=1}^{q}\text{slack}_{n_l}$ σαν μια επιπλέον πληροφορία που ανανεώνεται μετά από κάθε βήμα εισαγωγής και βήμα διαταραχής, οι μόνες ποσότητες που πρέπει να υπολογιστούν είναι το slack του κόμβου u_i, το maxStart που θα έχει ο κόμβος u_{n_j} μετά από μια πιθανή εισαγωγή του κόμβου u_i, όπως επίσης και το $\text{shift}_i^{n_j}$.

4.2.3 Ο αλγόριθμος AvgCSCRoutes

Ο αλγόριθμος AvgCSCRoutes βασίζεται στην ιδέα των Garcia et al. [31] της αναγωγής του TDTOPTW σε ένα TOPTW, της επίλυσης του δεύτερου με έναν γνωστό αλγόριθμο και μετά την εισαγωγή ενός βήματος επιδιόρθωσης για την μετατροπή των χρόνων σε χρονικά εξαρτημένους. Ο AvgCSCRoutes επεκτείνει αυτήν την ιδέα προσθέτοντας ακόμα ένα βήμα εισαγωγής που παίρνει υπ' όψιν του χρονικά εξαρτημένα κόστη ακμών. Ο αλγόριθμος διατυπώνεται συνοπτικά στα επόμενα 4 στάδια:

1. Για κάθε ακμή υπολογίζει το μέσο κόστος διαπέρασης.
2. Λύνει το TOPTW στιγμιότυπο, που προκύπτει αντικαθιστώντας τα χρονικά εξαρτημένα κόστη ακμών από τα μέσα κόστη, με τον CSCRoutes αλγόριθμο.
3. Επαναφέρει τα κόστη των ακμών της λύσης του TOPTW στα πραγματικά χρονικά κόστη που θα είχαν και επιδιορθώνει με ένα κατάλληλο βήμα μη έγκυρες διαδρομές.
4. Γεμίζει πιθανά χρονικά κενά στις διαδρομές χρησιμοποιώντας ένα σύντομο βήμα εισαγωγής, παίρνοντας υπ' όψιν τα χρονικά εξαρτημένα κόστη ακμών.

Το πρώτο στάδιο στον AvgCSCRoutes είναι ο υπολογισμός του μέσου κόστους διαπέρασης μιας ακμής. Δοθέντος του συνόλου των χρονικά εξαρτημένων κοστών διαπέρασης για κάθε μέρα της βδομάδας, ο μέσος χρόνος διαπέρασης υπολογίζεται ως

$$\text{avTr}_{ij} = \frac{\sum_{d=1}^{7}\sum_{t=0}^{1339}\text{travelling}_{ij}^{d}(t)}{7 \cdot 1440}$$

Έπειτα, ο αλγόριθμος CSCRoutes εκτελείται στο στιγμιότυπο του προβλήματος όπου τα κόστη των ακμών είναι πλέον ίσα με το μέσο κόστος που υπολογίστηκε προηγουμένως. Από την προηγούμενη εκτέλεση παράγεται ένα σύνολο από K διαδρομές, όπου όμως ο χρόνος μετάβασης μεταξύ διαδοχικών κορυφών δεν είναι ο πραγματικός. Για να είναι οι διαδρομές ρεαλιστικές, εφαρμόζονται τα χρονικά εξαρτημένα κόστη μετάβασης μεταξύ διαδοχικών κόμβων. Αυτό μπορεί να οδηγήσει σε μη έγκυρες λύσεις, είτε

επειδή πλέον κάποια διαδρομή θα έχει χρονικό κόστος μεγαλύτερο από το επιτρεπόμενο είτε γιατί η επίσκεψη κάποιου κόμβου θα γίνεται αργότερα από το επιτρεπόμενο (σύμφωνα με το χρονικό του παράθυρο). Σε περίπτωση που προκύψει κάποια διαδρομή μη έγκυρη, τότε εκτελείται επαναληπτικά ένα διορθωτικό βήμα έως ότου όλες οι διαδρομές γίνουν έγκυρες. Στο διορθωτικό βήμα αφαιρείται από την διαδρομή ο πρώτος (χρονικά) κόμβος που έχει χρόνο άφιξης μεγαλύτερο από τον χρόνο άφιξης στην λύση που έδωσε ο CSCRoutes (στην λύση με τα μέσα κόστη ακμών). Αν αυτός ο κόμβος είναι ο τερματικός κόμβος της διαδρομής, τότε δεν αφαιρείται εκείνος αλλά ο προηγούμενος του. Αυτό ολοκληρώνει το τρίτο στάδιο του αλγορίθμου. Λόγω της προηγούμενης διαδικασίας αρκετοί κόμβοι μπορεί να έχουν αφαιρεθεί από την λύση και έτσι να υπάρχει αρκετός ανεκμετάλευτος χρόνος. Για αυτόν τον λόγο εκτελείται το τέταρτο και τελευταίο στάδιο του αλγορίθμου. Στο τέταρτο στάδιο, ταξινομούνται οι κόμβοι που δεν έχουν εισαχθεί στην λύση σε φθίνουσα σειρά με βάση το κέρδος τους. Ύστερα, διαπερνώντας τους κόμβους με την σειρά, αν η εισαγωγή του υπό εξέταση κόμβου σε κάποια διαδρομή οδηγεί σε έγκυρη λύση, τότε ο κόμβος εισάγεται στην θέση με το μικρότερο shift (υπολογισμένο με τα πραγματικά χρονικά κόστη μετάβασης), αλλιώς αυτός ο κόμβος δεν εισάγεται στην λύση και εξετάζεται ο διαθέσιμος κόμβος με το αμέσως μικρότερο κέρδος.

4.3 Πειραματικά αποτελέσματα

4.3.1 Στιγμιότυπα

Το TDTOPTW είναι ένα πρόβλημα που εισήχθηκε πολύ πρόσφατα στην βιβλιογραφία. Για αυτόν τον λόγο δεν προϋπήρχαν δημοσιευμένα στιγμιότυπα αξιολόγησης αλγοριθμικών προσεγγίσεων για το TDTOPTW. Στην μόνη προϋπάρχουσα αλγοριθμική προσέγγιση του προβλήματος [31], χρησιμοποιήθηκαν στιγμιότυπα βασισμένα στην πόλη San Sebastian της Ισπανίας, που όμως δεν δημοσιεύτηκαν. Για τον λόγο αυτό δημιουργήθηκαν στιγμιότυπα για το TDTOPTW βασισμένα στην πόλη της Αθήνας.

Τα στιγμιότυπα περιλαμβάνουν 113 σημεία ενδιαφέροντος (POIs), π.χ. μουσεία, αρχαιολογικούς χώρους και πάρκα. Τα POIs βρίσκονται κυρίως στο κέντρο της Αθήνας και τον Πειραιά. Τα σημεία ενδιαφέροντος έχουν συλλεχθεί από διάφορες τουριστικές πύλες [1] και από υπηρεσίες διαδικτύου [2]. Το κέρδος της επίσκεψης σε κάθε POI παίρνει τιμή στο διάστημα $[1, 100]$ ανάλογη με την ευχαρίστηση που θα προσφέρει στον εκάστοτε περιηγητή. Ο χρόνος επίσκεψης στα αξιοθέατα κυμαίνεται από 1 λεπτό (π.χ. για κάποια πάρκα) έως 2 ώρες (για κάποια πολύ σημαντικά μουσεία και αρχαιολογικούς χώρους). Περίπου τα μισά από τα POIs θεωρούνται ανοικτά όλο το 24ωρο (π.χ. πάρκα) ενώ τα υπόλοιπα θεωρείται ότι έχουν ένα αρκετά μεγάλο καθημερινό χρονικό παράθυρο επίσκεψης (περίπου 8 ώρες).

Για την καλύτερη αξιολόγηση των αλγορίθμων έχουν δημιουργηθεί 3 διαφορετικές τοπολογίες (topol1, topol2, topol3), που περιέχουν τα ίδια POIs (ίδιες συντεταγμένες), με διαφορετικά όμως κέρδη, χρόνους επίσκεψης και χρονικά παράθυρα κόμβων ανά τοπολογία. Επιπλέον, για κάθε μία από τις τοπολογίες έχουν δημιουργηθεί 400 διαφορετικές πιθανές προτιμήσεις χρηστών, όπου αυτές συνοψίζονται σε 100 προτιμήσεις για κάθε πλήθος πιθανών διαδρομών από $k = 1$ έως 4. Κάθε προτίμηση περιλαμβάνει έναν αρχικό/τελικό κόμβο (που προσομοιώνει τον τόπο διαμονής) για κάθε μία από τις διαδρομές, όπως επίσης και ένα σύνολο κόμβων που δεν είναι επιθυμητή η επίσκεψη μια συγκεκριμένη ημέρα. Οι μη επιθυμητοί κόμβοι επιλέγονται ως εξής: (α) Κάθε κόμβος

[1] http://www.tripadvisor.com/, http://index.pois.gr/

[2] https://developers.google.com/places/documentation/

επιλέγεται να μην είναι επισκέψιμος κάθε μέρα του ταξιδιού με πιθανότητα 10% (αντιπροσωπεύοντας την δυνατότητα του τουρίστα να επιλέξει ότι δεν θέλει να επισκεφτεί τον συγκεκριμένο κόμβο). (β) Κάθε ένας από τους υπόλοιπους κόμβους, δεν είναι επισκέψιμος μια συγκεκριμένη μέρα με πιθανότητα 10% (θεωρώντας π.χ. ότι εκείνη την μέρα θα βρέχει και η επίσκεψη σε αυτόν τον κόμβο είναι αδύνατη). Τέλος, η ημερήσια επίσκεψη στα αξιοθέατα θεωρείται ότι γίνεται από τις 10:00-15:00.

Η μετακίνηση μεταξύ των POIs γίνεται χρησιμοποιώντας τα μέσα μαζικής μεταφοράς που προσφέρει το δίκτυο αστικών συγκοινωνιών της Αθήνας. Το δίκτυο αποτελείται από 3 γραμμές μετρό, 3 γραμμές τραμ και 287 γραμμές λεωφορείων έχοντας ένα σύνολο από 7825 σταθμούς. Ο Οργανισμός Αστικών Συγκοινωνιών Αθήνας ΟΑΣΑ παρείχε τα GTFS (General Transit Feed Specification) δεδομένα του συγκοινωνιακού δικτύου. Για την εκτέλεση των αλγορίθμων, για κάθε ζευγάρι από POIs έχει υπολογιστεί το σύνολο των μεγιστικών χρόνων (αναχώρησης,κόστους) (Ενότητα 4.1) χρησιμοποιώντας τον αλγόριθμο των Dibbelt et al. [24]. Τα στιγμιότυπα μπορούν να βρεθούν στο `http://www2.aegean.gr/dgavalas/public/tdtoptw_instances/index.html`.

4.3.2 Αποτελέσματα

Οι συγκρινόμενοι αλγόριθμοι είναι οι TDCSCRoutes, SlackCSCRoutes, AvgCSCRoutes που παρουσιάστηκαν στην προηγούμενη ενότητα και ο AvgILS που προτάθηκε από τους Garcia et al. [31]. Όλοι οι προτεινόμενοι αλγόριθμοι δημιουργούν στο προπαρασκευαστικό στάδιο $\lfloor \frac{N}{10} \rfloor = \lfloor \frac{113}{10} \rfloor = 11$ clusters. Η λίστα listOfClusterSets αποτελείται από $\lceil \text{numberOfClusters}/K \rceil$ μη επικαλυπτόμενες $K-$άδες από clusters, τυχαία επιλεγμένες. Ενώ, το μέγιστο πλήθος επαναλήψεων χωρίς βελτίωση της λύσης τίθεται ίσο με $\text{maxIterations} = \frac{400}{|\text{listOfClusterSets}|} \cdot \frac{K+1}{2 \cdot K}$. Επίσης, στους αλγορίθμους TDCSCRoutes και SlackCSCRoutes αποθηκεύεται το σύνολο των συντομότερων χρόνων μετάβασης μεταξύ όλων των POIs σε ένα $N \times N \times 1440 = 113 \times 113 \times 1440$ πίνακα που καταναλώνει περίπου 3.5 GB μνήμης. Τέλος, στον AvgCSCRoutes τα χρονικά εξαρτημένα κόστη των ακμών, που χρησιμοποιούνται στο 3ο και το 4ο βήμα του αλγορίθμου, υπολογίζονται με δυαδική αναζήτηση στα μεγιστικά ζευγάρια (αναχώρησης,κόστους).

Οι αλγόριθμοι έχουν υλοποιηθεί στην γλώσσα προγραμματισμού C++ και εκτελεστεί σε έναν προσωπικό υπολογιστή με επεξεργαστή Intel Core i5 στα 2.80 GHz και μνήμη RAM 4GB. Όλοι οι αλγόριθμοι έχουν εξεταστεί στα στιγμιότυπα που αναφέρθηκαν προηγουμένως.

Στους πίνακες 4.1 – 4.4 παρατίθενται τα αποτελέσματα των αλγορίθμων. Ονομαστικά, τα αποτελέσματα του TDCSCRoutes δίνονται στον Πίνακα 4.1, τα αποτελέσματα του SlackCSCRoutes παρουσιάζονται στον Πίνακα 4.2, τα αποτελέσματα για τον AvgCSCRoutes στον Πίνακα 4.3 ενώ τα αποτελέσματα για τον AvgILS παρατίθενται στον Πίνακα 4.4. Στους πίνακες περιέχονται οι εξής πληροφορίες: Για κάθε μία από τις τρεις τοπολογίες, η πρώτη στήλη περιγράφει το πλήθος των διαδρομών, στην δεύτερη στήλη εμφανίζεται το μέσο συλλεγμένο κέρδος από τις προτιμήσεις με αυτό το πλήθος των διαδρομών, ενώ στην τρίτη δίνεται ο μέσος χρόνος εκτέλεσης (σε ms). Στην τέταρτη εμφανίζεται το μέσο πλήθος των κόμβων που επισκέφτηκε η λύση, ενώ οι επόμενες τρεις στήλες δείχνουν το πλήθος των μετακινήσεων μεταξύ κόμβων, το πλήθος αυτών των μετακινήσεων στις οποίες χρησιμοποιήθηκαν μέσα μαζικής μεταφοράς και το ποσοστό των μετακινήσεων που έγινε με χρήση μέσων μαζικής μεταφοράς, αντίστοιχα. Η επόμενη στήλη δείχνει τον μέσο συνολικό χρόνο αναμονής στις λύσεις, δηλαδή το άθροισμα των χρόνων αναμονής στους κόμβους μέχρι να ξεκινήσει η επίσκεψη σε αυτούς (wait) και των χρόνων αναμονής έως ότου έρθει η επόμενη συγκοινωνία (delay). Ενώ η τελευταία στήλη αποτυπώνει την μέση αναμονή στους κόμβους έως την έλευση της επόμενης συγκοινωνίας (delay).

Πίνακας 4.1: Αποτελέσματα του αλγορίθμου TDCSCRoutes

K	Κέρδος	CPU(ms)	Επισκ	Μετακ	MMM	MMM(%)	Συν Αναμονή(min)	Μέση Αναμονή MMM
topol1								
1	1106.40	72.15	14.50	15.50	2.84	18.32	2.98	1.05
2	1772.25	189.05	24.48	26.48	4.66	17.60	5.35	1.15
3	2268.40	369.17	32.49	35.49	6.54	18.43	8.17	1.25
4	2670.15	586.82	39.88	43.88	8.20	18.69	11.71	1.43
topol2								
1	861.15	58.38	11.86	12.86	3.67	28.54	3.80	1.04
2	1520.80	161.47	21.81	23.81	6.17	25.91	7.59	1.23
3	2029.85	321.97	29.75	32.75	9.24	28.21	13.13	1.42
4	2475.10	589.65	37.30	41.30	12.02	29.10	17.19	1.43
topol3								
1	903.15	60.12	12.26	13.26	2.67	20.14	2.82	1.06
2	1553.25	166.25	21.52	23.52	5.77	24.53	7.82	1.36
3	2067.60	322.31	29.16	32.16	8.38	26.06	14.06	1.68
4	2515.35	537.34	36.71	40.71	10.66	26.19	18.84	1.77

Πίνακας 4.2: Αποτελέσματα του αλγορίθμου SlackCSCRoutes

K	Κέρδος	CPU(ms)	Επισκ	Μετακ	MMM	MMM(%)	Συν Αναμονή(min)	Μέση Αναμονή MMM
topol1								
1	1107.35	80.00	14.67	15.67	2.76	17.61	3.23	1.17
2	1756.65	201.62	24.78	26.78	4.40	16.43	5.96	1.35
3	2220.35	341.57	33.29	36.29	6.00	16.53	10.43	1.74
4	2618.80	511.73	41.49	45.49	7.44	16.36	14.12	1.90
topol2								
1	860.35	66.38	12.27	13.27	3.40	25.62	4.32	1.27
2	1512.05	176.66	22.42	24.42	6.07	24.86	9.40	1.55
3	2001.15	319.88	30.92	33.92	8.37	24.68	14.02	1.68
4	2429.40	478.68	38.97	42.97	11.21	26.09	19.94	1.78
topol3								
1	902.75	65.73	12.64	13.64	2.37	17.38	2.83	1.19
2	1538.05	177.23	21.93	23.93	5.37	22.44	9.68	1.80
3	2026.30	318.55	29.92	32.92	7.49	22.75	14.82	1.98
4	2450.35	502.50	37.78	41.78	10.32	24.70	20.32	1.97

Πίνακας 4.3: Αποτελέσματα του αλγορίθμου AvgCSCRoutes

K	Κέρδος	CPU(ms)	Επισκ	Μετακ	MMM	MMM(%)	Συν Αναμονή(min)	Μέση Αναμονή MMM
topol1								
1	1094.10	43.43	14.32	15.32	2.57	16.78	3.60	1.40
2	1757.40	109.31	24.45	26.45	4.30	16.26	6.86	1.60
3	2250.20	197.48	32.41	35.41	5.82	16.44	10.83	1.86
4	2661.70	314.77	39.78	43.78	7.17	16.38	14.17	1.98
topol2								
1	841.40	36.29	11.65	12.65	3.33	26.32	4.06	1.22
2	1500.95	92.91	21.60	23.60	5.57	23.60	8.60	1.54
3	2010.45	177.22	29.85	32.85	7.35	22.37	12.61	1.72
4	2452.35	290.35	37.29	41.29	10.34	25.04	22.51	2.18
topol3								
1	888.60	39.02	12.19	13.19	2.25	17.06	2.66	1.18
2	1530.65	100.42	21.51	23.51	4.76	20.25	8.85	1.86
3	2044.90	180.14	29.41	32.41	6.22	19.19	12.74	2.05
4	2487.65	291.94	36.94	40.94	8.03	19.61	16.62	2.07

Πίνακας 4.4: Αποτελέσματα του αλγορίθμου AvgILS

K	Κέρδος	CPU(ms)	Επισκ	Μετακ	ΜΜΜ	ΜΜΜ(%)	Συν Αναμονή(min)	Μέση Αναμονή ΜΜΜ
topol1								
1	1079.85	64.80	13.85	14.85	2.50	16.84	4.81	1.92
2	1748.05	213.70	23.90	25.90	4.17	16.10	7.79	1.87
3	2236.05	406.79	31.86	34.86	5.75	16.49	12.25	2.13
4	2645.20	505.49	39.07	43.07	7.21	16.74	15.62	2.17
topol2								
1	823.85	50.11	11.07	12.07	3.08	25.52	4.99	1.62
2	1478.35	183.83	21.08	23.08	5.41	23.44	9.52	1.76
3	1988.85	336.31	29.10	32.10	7.24	22.55	15.15	2.09
4	2424.05	523.93	36.30	40.30	10.22	25.36	24.06	2.35
topol3								
1	871.70	54.02	11.78	12.78	2.30	18.00	4.01	1.74
2	1505.60	195.03	20.79	22.79	4.53	19.88	9.62	2.12
3	2017.00	357.91	28.49	31.49	6.04	19.18	13.85	2.29
4	2454.05	491.05	35.81	39.81	8.01	20.12	19.52	2.44

Στον Πίνακα 4.5 αναλύονται τα συγκριτικά αποτελέσματα των αλγορίθμων με βάση το πλήθος των διαδρομών, το συλλεγμένο κέρδος, τον χρόνο εκτέλεσης, το πλήθος των επισκέψεων και το πλήθος των μετακινήσεων με χρήση μέσων μαζικής μεταφοράς. Για κάθε αλγόριθμο και κάθε πιθανό πλήθος διαδρομών ($1 \leq K \leq 4$) θεωρείται η μέση τιμή που θα προκύψει από κάθε μία από τις 100 προτιμήσεις για κάθε μία από τις τρεις τοπολογίες (topol1, topol2, topol3). Κάθε μία ποσότητα κλιμακώνεται ανάλογα με την μέγιστη ποσότητα που θα φέρει κάποιος αλγόριθμος. Για παράδειγμα, για 1 διαδρομή το μέγιστο μέσο πλήθος επισκέψεων έχει ο SlackCSCRoutes οπότε παίρνει την τιμή 100 και ο TDCSCRoutes έχει πλήθος επισκέψεων ίσο με 97.57 % των επισκέψεων του SlackCSCRoutes. Οι τιμές που είναι γραμμένες με έντονα γράμματα αποτυπώνουν την καλύτερη απόδοση που είχε κάποιος αλγόριθμος για την συγκεκριμένη παράμετρο.

Όπως προκύπτει από τον πίνακα, ο αλγόριθμος που παράγει τις λύσεις με το μεγαλύτερο κέρδος είναι ο TDCSCRoutes. Ο TDCSCRoutes υπερέχει των υπόλοιπων αλγορίθμων σε κέρδος για όλα τα πιθανά πλήθη διαδρομών. Ενώ, έχει χρόνο εκτέλεσης παρόμοιο με τον SlackCSCRoutes.

Ο αλγόριθμος με τον λιγότερο χρόνο εκτέλεσης είναι ο AvgCSCRoutes. Ο αλγόριθμος είναι περίπου μισή φορά γρηγορότερος από ότι ο SlackCSCRoutes και λίγο λιγότερο από τους υπόλοιπους. Εκτός από την πολύ καλή ταχύτητά του, ο AvgCSCRoutes παράγει λύσεις υψηλής ποιότητας με κέρδος που απέχει κατά μέσο όρο το πολύ 2% από το κέρδος των λύσεων που δίνει ο TDCSCRoutes. Πολύ θετικό στοιχείο για τον AvgCSCRoutes είναι ότι υπερτερεί του AvgILS και στον χρόνο εκτέλεσης και στο συλλεγμένο κέρδος. Αυτό είναι σημαντικό καθώς αυτοί είναι οι δύο αλγόριθμοι που δεν λύνουν απευθείας το TDTOPTW, αλλά χρησιμοποιώντας τα μέσα χρονικά κόστη λύνουν ένα στιγμιότυπο του TOPTW και επεκτείνουν την λύση που θα βρουν σε λύση για το TDTOPTW. Η υπεροχή αυτή του AvgCSCRoutes όσον αφορά το κέρδος βασίζεται κυρίως στο τελευταίο βήμα εισαγωγής που θεωρεί τα χρονικά εξαρτημένα κόστη ακμών. Τέλος, τα αποτελέσματα του AvgCSCRoutes είναι αρκετά κοντά και στο βέλτιστο παραγόμενο πλήθος επισκέψεων (απόκλιση < 4%) και στο βέλτιστο πλήθος μετακινήσεων με μέσα μαζικής μεταφοράς (απόκλιση < 3%).

Ο αλγόριθμος με τις ελάχιστες μετακινήσεις με μέσα μαζικής μεταφοράς είναι ο AvgILS. Αυτό οφείλεται κυρίως στο ότι είναι ο μόνος αλγόριθμος ο οποίος σε καμιά φάση της εκτέλεσής του δεν παίρνει υπ' όψιν τα χρονικά εξαρτημένα κόστη ακμών για την εισαγωγή νέων κόμβων στην λύση. Οπότε, όταν τα κόστη των ακμών παίρνουν τις πραγματικές τους τιμές στο επιδιορθωτικό βήμα, είναι πολύ πιθανό η συντομότερη διαδρομή μεταξύ δύο διαδοχικών κόμβων σε κάποια διαδρομή να γίνεται με

πεζοπορία. Επιπλέον, επειδή το μέσο κόστος των ακμών είναι συνήθως μεγαλύτερο από το ελάχιστο χρονικό κόστος μετάβασης μια καθορισμένη στιγμή αναχώρησης, η δυνατότητα εισαγωγής νέων κόμβων περιορίζεται. Αυτό έχει ως αποτέλεσμα ο AvgILS να είναι ο αλγόριθμος με τις λιγότερες επισκέψεις σε αξιοθέατα.

Τέλος, ο αλγόριθμος SlackCSCRoutes παράγει τις λύσεις με το μεγαλύτερο πλήθος επισκεπτόμενων κόμβων. Αυτό συμβαίνει καθώς ο αλγόριθμος δημιουργεί μεγάλα slacks στους επισκεπτόμενους κόμβους, ευνοώντας την εισαγωγή πολλών νέων κόμβων. Ο αλγόριθμος επίσης παράγει λύσεις υψηλού κέρδους που υπολείπονται πολύ λίγο του TDCSCRoutes. Για παράδειγμα για 1 διαδρομή οι δύο αλγόριθμοι έχουν σχεδόν ίδιο μέσο κέρδος, ενώ όσο το πλήθος των διαδρομών αυξάνεται η διαφορά μειώνεται χωρίς ωστόσο ποτέ να ξεπεράσει το 3%.

Πίνακας 4.5: Συγκριτικά Αποτελέσματα των Αλγορίθμων

K		Κέρδος	CPU(ms)	Επισκέψεις	Χρήση MMM
1	TDCSCRoutes	**100**	89.88	97.57	100
	SlackCSCRoutes	99.99	100	**100**	92.92
	AvgCSCRoutes	98.38	**55.98**	96.41	88.78
	AvgILS	96.68	79.64	92.72	**85.84**
2	TDCSCRoutes	**100**	87.21	98.09	100
	SlackCSCRoutes	99.18	93.75	**100**	95.42
	AvgCSCRoutes	98.82	**51.07**	97.73	88.13
	AvgILS	97.64	100	95.14	**85**
3	TDCSCRoutes	**100**	92.05	97.1	100
	SlackCSCRoutes	98.15	89.01	**100**	90.48
	AvgCSCRoutes	99.05	**50.39**	97.39	80.26
	AvgILS	98.05	100	95.03	**78.77**
4	TDCSCRoutes	**100**	100	96.32	100
	SlackCSCRoutes	97.88	87.11	**100**	93.81
	AvgCSCRoutes	99.23	**52.34**	96.42	82.71
	AvgILS	98.21	88.72	94.03	**82.38**

4.4 Συμπεράσματα

Στο κεφάλαιο αυτό προτάθηκαν τρεις νέοι ευρετικοί αλγόριθμοι για το TDTOPTW. Ονομαστικά, οι αλγόριθμοι που προτάθηκαν είναι οι TDCSCRoutes, SlackCSCRoutes και AvgCSCRoutes και βασίζονται στον αλγόριθμο CSCRoutes για το TOPTW. Οι προτεινόμενοι αλγόριθμοι μπορούν να χρησιμοποιηθούν για τον σχεδιασμό τουριστικών διαδρομών σε πόλεις, όπου οι μετακινήσεις γίνονται με χρήση μέσων μαζικής μεταφοράς. Οι στόχοι των αλγορίθμων ήταν τρεις:

- Η παραγωγή πολύ ευχάριστων διαδρομών για τους τουρίστες
- Η παραγωγή διαδρομών με όσο το δυνατόν λιγότερες μετακινήσεις με χρήση μέσων μαζικής μεταφοράς, ευνοώντας ταυτόχρονα το περπάτημα.
- Ο γρήγορος χρόνος εκτέλεσης των αλγορίθμων καθώς η χρήση τους είναι επιθυμητό να γίνει σε εφαρμογές διαδικτύου.

Οι τρεις αλγόριθμοι μαζί με τον αλγόριθμο AvgILS των Garcia et al. [31] αξιολογήθηκαν σε καινούργια στιγμιότυπα δημιουργημένα με βάση την πόλη της Αθήνας, όπου οι μετακινήσεις μεταξύ των αξιοθεάτων γίνονται με χρήση των αστικών συγκοινωνιών. Τα αποτελέσματα έδειξαν ότι ο αλγόριθμος που παράγει τις καλύτερες λύσεις από άποψη κέρδους είναι ο TDCSCRoutes. Ο SlackCSCRoutes παράγει τις λύσεις με τις περισσότερες επισκέψεις σε αξιοθέατα παρουσιάζοντας καλή ισορροπία στα υπόλοιπα

κριτήρια της σύγκρισης. Ο AvgILS παρουσιάζει τις λιγότερες μετακινήσεις με χρήση μέσων μαζικής μεταφοράς, όμως με τις λιγότερος επισκέψεις σε αξιοθέατα. Τέλος, ο AvgCSCRoutes είναι ο πιο αποδοτικός αλγόριθμος με χρόνο εκτέλεσης σχεδόν τον μίσο από τους προηγούμενους αλγορίθμους διατηρώντας αρκετά καλή απόδοση και στις υπόλοιπες παραμέτρους που πάρθηκαν υπ' όψιν.

Τα αποτελέσματα έδειξαν ότι δεν υπάρχει κάποιος αλγόριθμος που να υπερτερεί σε όλα τα κριτήρια, οπότε η επιλογή του αλγορίθμου που θα χρησιμοποιηθεί βασίζεται στους στόχους της εκάστοτε εφαρμογής. Αν η εφαρμογή στοχεύει στην εύρεση πολύ κερδοφόρων διαδρομών, επιτρέποντας ένα αυξημένο χρόνο εκτέλεσης, τότε ο πιο κατάλληλος αλγόριθμος είναι ο TDCSCRoutes. Αντίθετα, αν η εφαρμογή έχει έναν πολύ περιορισμένο χρόνο εκτέλεσης επιτρέποντας μικρή απώλεια στο συλλεγμένο κέρδος τότε ο πιο κατάλληλος αλγόριθμος είναι ο AvgCSCRoutes.

Κεφάλαιο 5

Συμπεράσματα

Στην παρούσα εργασία μελετήθηκε το Πρόβλημα του Σχεδιασμού Τουριστικών Διαδρομών (TTDP). Το TTDP είναι το πρόβλημα που αντιμετωπίζει ένας τουρίστας όταν επισκέπτεται μια τουριστική περιοχή, έχοντας να σχεδιάσει τις διαδρομές που θα ακολουθήσει κάθε μέρα για να δει τα πιο ταιριαστά σε εκείνον αξιοθέατα. Το TTDP μοντελοποιείται συνήθως ως OP ή μια επέκταση του. Δυο διαφορετικές μοντελοποιήσεις του προβλήματος εξετάστηκαν, ονομαστικά το TOPTW και το TDTOPTW. Και για τις δύο μοντελοποιήσεις προτάθηκαν νέες αλγοριθμικές προσεγγίσεις. Στόχος των προσεγγίσεων αυτών ήταν (i) η παραγωγή πολύ “ευχάριστων” διαδρομών για τον τουρίστα, (ii) η παραγωγή διαδρομών με λίγες μετακινήσεις με μέσα μαζικής μεταφοράς και ταυτόχρονα αρκετές μετακινήσεις με πεζοπορία και (iii) ο γρήγορος χρόνος εκτέλεσης των αλγορίθμων, ώστε να μπορούν να χρησιμοποιηθούν σε εφαρμογές διαδικτύου.

Στο Κεφάλαιο 3 εισήχθηκαν δυο καινούργιοι ευρετικοί αλγόριθμοι για το TOPTW, ο CSCRatio και ο CSCRoutes. Οι αλγόριθμοι βασίζονται στον πιο κατάλληλο για εφαρμογές διαδικτύου αλγόριθμο, τον ILS, και βελτιώνουν τις αδυναμίες του δημιουργώντας ομάδες από κοντινά αξιοθέατα. Οι τρεις αλγόριθμοι συγκρίθηκαν σε προϋπάρχοντα δημοσιευμένα στιγμιότυπα, όπως επίσης και σε νέα, δημιουργημένα ώστε να προσομοιώνουν ρεαλιστικές τουριστικές τοποθεσίες. Από την σύγκριση προέκυψε ότι ο CSCRatio παράγει τις πιο κερδοφόρες διαδρομές, ενώ υπερτερεί του ILS στις μετακινήσεις μεταξύ μακρινών κόμβων έχοντας ταυτόχρονα συγκρίσιμο χρόνο εκτέλεσης. Από την άλλη μεριά, ο CSCRoutes παράγει τις λιγότερο κερδοφόρες λύσεις, όμως με τις λιγότερες μετακινήσεις μεταξύ των ομάδων των αξιοθεάτων και ταυτόχρονα όντας ο πιο γρήγορος αλγόριθμος.

Στο Κεφάλαιο 4 προτάθηκαν τρεις νέοι ευρετικοί αλγόριθμοι για το TDTOPTW, οι TDCSCRoutes, SlackCSCRoutes και AvgCSCRoutes. Οι αλγόριθμοι επεκτείνουν τον CSCRoutes ώστε να μπορεί να χειριστεί χρονικά εξαρτημένα κόστη ακμών. Οι αλγόριθμοι συγκρίθηκαν με τον αλγόριθμο AvgILS, που αποτελεί την μοναδική προϋπάρχουσα τεχνική για την επίλυση αυθαίρετων στιγμιοτύπων του TDTOPTW. Η σύγκριση πραγματοποιήθηκε σε νέα στιγμιότυπα που δημιουργήθηκαν με βάση την Αθήνα. Ο TDCSCRoutes παρήγαγε τις πιο κερδοφόρες λύσεις, ενώ ο SlackCSCRoutes τις λύσεις με τις περισσότερες επισκέψεις σε αξιοθέατα. Ακόμα, ο AvgILS παρήγαγε τις λύσεις με τις λιγότερες μετακινήσεις με χρήση μέσων μαζικής μεταφοράς. Τέλος, ο AvgCSCRoutes ήταν αρκετά πιο γρήγορος από τους υπόλοιπους αλγορίθμους, ενώ υπολειπόταν ελάχιστα στα υπόλοιπα εξεταζόμενα κριτήρια.

Βιβλιογραφία

[1] R. A. Abbaspour and F. Samadzadegan. Time-dependent personal tour planning and scheduling in metropolises. *Expert Systems and Applications*, 38:12439–12452, 2011.

[2] C. Archetti, A. Hertz, and M. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13:49–76, 2007.

[3] E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, SCG '98, pages 307–316, 1998.

[4] M. Bagirov. Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition*, 41(10):3192 – 3199, 2008.

[5] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 166–174, 2004.

[6] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):pp. 316–329, 1998.

[7] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science*, pages 46 –55, 2003.

[8] A. Blum, S. Chawla, D. R. Karger, Lane T., A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM J. Comput.*, 37(2):653–670, 2007.

[9] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35:268–308, September 2003.

[10] H. Bouly, D.-C. Dang, and A. Moukrim. A memetic algorithm for the team orienteering problem. *4OR: A Quarterly Journal of Operations Research*, 8:49–70, 2010.

[11] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, 5:211–230, 2007.

[12] S. E. Butt and T. M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101 – 111, 1994.

[13] S. E. Butt and D. M. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers and Operations Research*, 26(4):427 – 441, 1999.

[14] I-M. Chao, B. L. Golden, and E. A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.

[15] I-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[16] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *Proceedings of the 19th Annual ACM-SIAM symposium on Discrete Algorithms*, SODA '08, pages 661–670, 2008.

[17] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *Proceedings of Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques*, pages 72–83, 2004.

[18] C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *Proceedings of the 46th Annual IEEE Symposium on the Foundations of Computer Science*, pages 245 – 253, 2005.

[19] K. Chen and S. Har-Peled. The orienteering problem in the plane revisited. In *Proceedings of the 22nd Annual Symposium on Computational Geometry*, SCG '06, pages 247–254, 2006.

[20] K. Cheverst, K. Mitchell, and N. Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Communications of the ACM*, 45(5):47–51, 2002.

[21] City trip planner, http://www.citytripplanner.com/, Last accessed: March 2014.

[22] J-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

[23] D.-C. Dang, R. N. Guibadj, and A. Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332 – 344, 2013.

[24] J. Dibbelt, T. Pajor, and D. Wagner. User-constrained multi-modal route planning. In *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*, pages 118 – 129, 2012.

[25] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.

[26] M. Fischetti, J. J. S. González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

[27] F. V. Fomin and A. Lingas. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83(2):57 – 62, 2002.

[28] G. Frederickson and B. Wittman. Approximation algorithms for the traveling repairman and speeding deliveryman problems. *Algorithmica*, 62:1198–1221, 2012.

[29] A. Garcia, O. Arbelaitz, M. Linaza, P. Vansteenwegen, and W. Souffriau. Personalized tourist route generation. In F. Daniel and F. Facca, editors, *Current Trends in Web Engineering*, volume 6385 of *Lecture Notes in Computer Science*, pages 486–497, 2010.

[30] A. Garcia, M. Linaza, O. Arbelaitz, and P. Vansteenwegen. Intelligent routing system for a personalised electronic tourist guide. In W. Höpken, U. Gretzel, and R. Law, editors, *Information and Communication Technologies in Tourism 2009*, pages 185–197, 2009.

[31] A. Garcia, P. Vansteenwegen, O. Arbelaitz, W. Souffriau, and M. T. Linaza. Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, 40(3):758 – 774, 2013.

[32] D. Gavalas, M. Kenteris, C. Konstantopoulos, and G. Pantziou. Web application for recommending personalised mobile tourist routes. *IET Software*, 6(4):313–322, 2012.

[33] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 2014. in press.

[34] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and Y. Tasoulas. Cluster-based heuristics for the team orienteering problem with time windows. In *Proceedings of 12th International Symposium on Experimental Algorithms (SEA'13)*, pages 390–401, 2013.

[35] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis. Efficient heuristics for the time dependent team orienteering problem with time windows. In *Proceedings of the 1st International Conference on Applied Algorithms*, pages 152–163, 2014.

[36] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32(4):263–273, 1998.

[37] M. Gendreau, G. Laporte, and F. Semet. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2-3):539 – 545, 1998.

[38] B. Golden, Q. Wang, and L. Liu. A multifaceted heuristic for the orienteering problem. *Naval Research Logistics*, 35(3):359–366, 1988.

[39] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[40] Q. Hu and A. Lim. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276 – 286, 2014.

[41] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[42] M. G. Kantor and M. B. Rosenwein. The orienteering problem with time windows. *The Journal of the Operational Research Society*, 43(6):pp. 629–635, 1992.

[43] S. Kataoka and S. Morito. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31(4):515–530, 1988.

[44] L. Ke, C. Archetti, and Z. Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3):648–665, 2008.

[45] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[46] M. Kenteris, D. Gavalas, and D. Economou. An innovative mobile electronic tourist guide application. *Personal and Ubiquitous Computing*, 13:103–118, 2009.

[47] M. Kenteris, D. Gavalas, and D. Economou. Electronic mobile guides: a survey. *Personal and Ubiquitous Computing*, 15:97–111, 2011.

[48] N. J. Korula. *Approximation algorithms for network design and orienteering*. PhD thesis, University of Illinois at Urbana-Champaign, 2010.

[49] N. Labadi, R. Mansini, J. Melechovský, and R. Wolfler Calvo. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15 – 27, 2012.

[50] N. Labadi, J. Melechovský, and R. Calvo. An effective hybrid evolutionary local search for orienteering and team orienteering problems with time windows. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 219–228, 2010.

[51] N. Labadi, J. Melechovský, and R. Wolfler Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17:729–753, 2011.

[52] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358, 1992.

[53] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2-3):193 – 207, 1990.

[54] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[55] J. Li. Model and algorithm for time-dependent team orienteering problem. In S. Lin and X. Huang, editors, *Advanced Research on Computer Education, Simulation and Modeling*, volume 175 of *Communications in Computer and Information Science*, pages 1–7. Springer Berlin Heidelberg, 2011.

[56] J. Li, Q. Wu, X. Li, and D. Zhu. Study on the time-dependent orienteering problem. In *Proceedings of the 2010 International Conference on E-Product E-Service and E-Entertainment (ICEEE'2010)*, pages 1–4, 2010.

[57] A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451 – 461, 2003.

[58] S.-W. Lin and V. F. Yu. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94 – 107, 2012.

[59] H. Lourenco, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer New York, 2003.

[60] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[61] John E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P.M. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization*, page 65–77. Oxford University Press, 2002.

[62] R. Montemanni and L. M. Gambardella. An ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, 34(4):287–306, 2009.

[63] mtrip travel guides, http://www.mtrip.com/, Last accessed: March 2014.

[64] S. Muthuswamy and S. Lam. Discrete particle swarm optimization for the team orienteering problem. *Memetic Computing*, 3:287–303, 2011.

[65] V. Nagarajan and R. Ravi. The directed orienteering problem. *Algorithmica*, 60:1017–1030, August 2011.

[66] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[67] C.H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[68] J.M. Peña, Lozano J.A., and Larrañaga P. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027 – 1040, 1999.

[69] R. Ramesh and K. M. Brown. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151 – 165, 1991.

[70] R. Ramesh, Y.-S. Yoon, and M. H. Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2):155–165, 1992.

[71] G. Righini and M. Salani. Dynamic programming for the orienteering problem with time windows, March 2006.

[72] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191 – 1203, 2009.

[73] M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35:254–265, 1987.

[74] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. A greedy randomised adaptive search procedure for the team orienteering problem. In *EU/MEeting 2008 on metaheuristics for logistics and vehicle routing*, 2008.

[75] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37(11):1853 – 1859, 2010.

[76] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379 – 1407, 2005.

[77] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[78] P. Vansteenwegen. *Planning in Tourism and Public Transportation - Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling*. PhD thesis, Katholieke Universiteit Leuven, 2008.

[79] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118 – 127, 2009.

[80] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36:3281–3290, 2009.

[81] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. The city trip planner: An expert system for tourists. *Expert Systems with Applications*, 38(6):6540 – 6546, 2011.

[82] P. Vansteenwegen and D. Van Oudheusden. The mobile tourist guide: An or opportunity. *Operational Research Insight*, 20(3):21–27, 2007.

[83] Pieter Vansteenwegen, Wouter Souffriau, GreetVanden Berghe, and DirkVan Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the Service Industry*, volume 624 of *Lecture Notes in Economics and Mathematical Systems*, pages 15–31. Springer Berlin Heidelberg, 2009.

[84] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[85] Q. Wang, X. Sun, B. L. Golden, and J. Jia. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61:111–120, 1995.

[86] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 1st edition, 2011.

[87] Β. Ζησιμόπουλος. *Αλγοριθμική Επιχειρησιακή Έρευνα*. Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, 2007.

Παράρτημα Α΄

Αναλυτικά αποτελέσματα των αλγορίθμων για το TOPTW

Πίνακας Α′.1: Αποτελέσματα για τα στιγμιότυπα του Solomon για 1 διαδρομή

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
c101	10	320	7	104	310	8	94	300	5	92
c102	10	360	6	95	360	6	128	360	6	111
c103	10	390	8	101	390	8	201	380	8	159
c104	10	400	9	128	420	7	217	410	7	205
c105	10	340	9	97	340	11	122	330	7	99
c106	10	340	8	101	340	9	130	330	6	124
c107	10	360	8	104	360	8	142	360	6	113
c108	10	370	8	124	370	8	158	360	6	129
c109	10	380	8	111	380	6	172	380	6	144
c201	10	840	16	553	860	14	370	840	10	161
c202	10	910	14	780	910	12	469	890	9	146
c203	10	940	19	726	940	14	647	900	10	169
c204	10	950	17	549	960	11	843	970	10	235
c205	10	900	13	406	900	12	476	890	10	167
c206	10	910	15	419	920	12	513	900	10	190
c207	10	910	17	623	930	12	570	920	10	203
c208	10	930	14	463	930	13	618	920	10	201
r101	10	182	6	61	183	7	64	180	5	70
r102	10	286	6	111	286	6	122	282	5	104
r103	10	286	7	101	291	6	166	289	6	127
r104	10	297	6	117	301	4	167	303	5	151
r105	10	247	5	135	247	5	91	238	3	85
r106	10	293	6	108	293	6	128	279	5	116
r107	10	288	7	100	294	5	162	289	6	119
r108	10	297	5	170	308	5	187	303	5	146
r109	10	276	7	124	276	7	112	259	3	95
r110	10	281	4	144	281	4	133	281	4	109
r111	10	295	6	129	295	6	160	297	4	119
r112	10	295	6	114	295	6	185	285	3	132
r201	10	788	28	709	786	23	491	476	10	115
r202	10	880	27	965	891	21	654	788	10	161
r203	10	980	22	1781	983	22	975	914	10	231
r204	10	1073	23	909	1057	16	1380	1048	10	291
r205	10	931	28	1452	905	28	815	644	10	144
r206	10	996	21	701	996	22	915	849	10	206
r207	10	1038	21	789	1059	20	1101	917	10	259
r208	10	1069	17	1524	1083	17	1437	1061	10	297
r209	10	926	22	706	920	23	963	717	10	221
r210	10	958	24	1132	970	21	976	813	10	200
r211	10	1023	24	728	1025	17	1216	865	10	274
rc101	10	219	4	93	219	4	78	219	4	79
rc102	10	259	5	117	259	5	98	266	4	81
rc103	10	265	6	100	263	4	115	266	4	95
rc104	10	297	5	83	301	4	133	301	4	110
rc105	10	221	4	116	244	4	93	241	4	86
rc106	10	239	5	124	250	4	95	250	4	86
rc107	10	274	5	124	276	5	110	261	4	102
rc108	10	288	5	104	288	5	125	274	4	111
rc201	10	780	19	583	777	19	384	646	10	121
rc202	10	882	19	762	924	14	543	864	10	165
rc203	10	960	13	761	956	18	750	901	10	173
rc204	10	1117	14	852	1108	11	1100	1121	10	279
rc205	10	840	17	564	845	15	435	685	10	138
rc206	10	860	19	541	869	18	568	751	10	140
rc207	10	926	17	896	925	15	771	801	10	205
rc208	10	1037	17	1226	1026	16	884	971	10	254

Πίνακας Α′.2: Αποτελέσματα για τα στιγμιότυπα του Solomon για 2 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
c101	10	590	15	278	590	12	195	550	8	141
c102	10	650	11	410	650	14	255	650	11	220
c103	10	700	14	305	710	13	407	700	12	310
c104	10	750	14	534	750	14	506	740	12	408
c105	10	640	16	288	640	16	218	600	12	196
c106	10	620	17	282	620	17	263	600	12	241
c107	10	670	13	304	670	13	259	620	11	184
c108	10	670	16	299	670	16	311	640	9	197
c109	10	710	10	378	720	10	398	690	9	289
c201	10	1400	24	1115	1420	21	838	1430	15	375
c202	10	1430	29	873	1430	23	1164	1440	18	385
c203	10	1430	28	848	1440	25	1365	1440	15	399
c204	10	1460	28	1080	1450	26	1491	1460	15	466
c205	10	1450	18	1791	1450	18	1098	1440	17	487
c206	10	1440	17	923	1470	15	1304	1470	14	450
c207	10	1450	22	1078	1470	16	1184	1470	14	504
c208	10	1460	19	1178	1480	17	1478	1460	16	540
r101	10	330	12	180	343	10	124	325	7	107
r102	10	508	11	290	501	12	255	501	9	174
r103	10	513	13	292	514	12	327	504	8	258
r104	10	539	10	346	543	10	397	529	10	287
r105	10	430	10	252	442	10	171	422	7	153
r106	10	529	12	411	524	12	256	505	10	199
r107	10	529	11	332	524	10	310	523	10	252
r108	10	549	11	345	556	9	386	552	9	308
r109	10	498	11	376	506	12	258	480	7	173
r110	10	515	9	455	508	8	276	506	7	255
r111	10	535	10	605	538	10	326	538	10	217
r112	10	515	10	523	538	9	423	531	7	312
r201	10	1231	54	838	1212	46	1299	864	19	269
r202	10	1270	47	955	1302	44	1285	1115	20	447
r203	10	1377	46	726	1372	40	1304	1243	20	560
r204	10	1440	38	565	1438	32	1453	1364	19	637
r205	10	1338	40	913	1333	39	1390	1115	20	413
r206	10	1401	44	667	1406	41	1672	1294	20	458
r207	10	1428	48	588	1434	44	1574	1378	20	758
r208	10	1458	43	456	1458	36	1667	1419	20	597
r209	10	1345	48	933	1370	40	1702	1187	19	429
r210	10	1365	42	915	1383	41	1438	1281	20	465
r211	10	1422	37	656	1438	35	2073	1316	18	552
rc101	10	427	7	506	427	7	150	419	8	130
rc102	10	494	7	355	488	8	244	497	7	147
rc103	10	519	7	275	516	8	262	519	8	192
rc104	10	565	9	481	574	8	324	555	7	277
rc105	10	459	8	327	478	8	181	435	6	168
rc106	10	458	11	394	481	8	254	464	8	174
rc107	10	515	9	488	514	8	258	487	7	255
rc108	10	546	11	388	536	10	296	535	8	216
rc201	10	1305	41	865	1343	37	891	1034	19	311
rc202	10	1461	34	1055	1435	37	1011	1184	18	362
rc203	10	1573	35	846	1562	32	1298	1364	19	396
rc204	10	1656	26	654	1666	26	1445	1607	19	469
rc205	10	1381	36	1136	1363	34	1012	1045	17	359
rc206	10	1495	34	820	1477	34	1288	1326	20	605
rc207	10	1531	29	873	1508	31	1518	1427	19	483
rc208	10	1606	31	1389	1610	29	1579	1583	19	579

Πίνακας Α′.3: Αποτελέσματα για τα στιγμιότυπα του Solomon για 3 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
c101	10	790	20	704	800	22	310	780	15	308
c102	10	890	20	664	890	19	419	880	15	541
c103	10	960	20	772	960	17	626	940	16	534
c104	10	1010	15	960	1010	16	981	1020	14	760
c105	10	840	26	615	850	20	398	830	16	361
c106	10	840	23	764	850	23	447	830	16	441
c107	10	900	19	1134	900	19	464	880	15	354
c108	10	900	22	1875	910	23	514	880	13	482
c109	10	950	16	674	950	15	694	960	15	737
c201	10	1750	35	1176	1760	27	1341	1750	22	618
c202	10	1750	38	947	1780	28	1151	1750	27	548
c203	10	1760	44	584	1750	35	1247	1750	23	583
c204	10	1780	43	525	1780	30	1579	1790	22	653
c205	10	1770	26	636	1770	24	1330	1790	21	715
c206	10	1770	21	538	1800	21	1523	1800	20	595
c207	10	1810	22	875	1790	24	1463	1780	20	641
c208	10	1810	21	883	1810	21	1721	1810	19	756
r101	10	481	17	314	475	16	167	448	11	163
r102	10	685	14	537	670	16	412	666	13	334
r103	10	720	15	878	720	14	548	708	12	365
r104	10	765	13	1138	767	15	621	746	11	488
r105	10	609	19	831	596	16	294	592	11	289
r106	10	719	13	594	704	13	403	699	13	453
r107	10	747	14	810	743	14	540	744	13	471
r108	10	790	14	1616	794	14	671	769	13	518
r109	10	699	16	1110	697	16	495	677	12	325
r110	10	711	16	694	718	16	538	707	13	491
r111	10	764	14	941	764	14	602	758	12	466
r112	10	758	14	1023	757	14	864	746	13	598
r201	10	1408	65	784	1412	57	1352	1170	29	478
r202	10	1443	59	619	1443	58	1414	1344	27	601
r203	10	1458	63	394	1458	54	1494	1387	30	705
r204	10	1458	56	311	1458	40	1711	1444	24	766
r205	10	1458	63	408	1458	60	1379	1404	29	656
r206	10	1458	52	334	1458	54	1523	1428	27	778
r207	10	1458	63	318	1458	58	1570	1453	27	654
r208	10	1458	49	309	1458	39	1673	1458	24	729
r209	10	1458	53	359	1458	52	1443	1409	28	599
r210	10	1458	56	347	1458	51	1455	1420	28	611
r211	10	1458	56	315	1458	44	1508	1458	27	607
rc101	10	604	12	592	614	9	236	614	9	216
rc102	10	698	13	1389	695	11	459	692	12	409
rc103	10	747	11	760	763	12	500	729	11	359
rc104	10	822	11	687	816	11	512	804	10	523
rc105	10	654	11	487	648	11	303	634	9	271
rc106	10	678	13	549	683	13	390	670	11	358
rc107	10	745	14	690	752	15	564	724	11	350
rc108	10	757	14	562	780	12	531	763	11	538
rc201	10	1625	53	743	1650	53	1385	1374	27	564
rc202	10	1686	58	664	1684	49	1530	1465	26	713
rc203	10	1724	42	444	1724	43	1398	1612	25	885
rc204	10	1724	42	350	1724	40	1534	1701	26	634
rc205	10	1659	54	601	1660	49	1462	1376	28	551
rc206	10	1708	43	562	1721	47	1734	1583	26	706
rc207	10	1713	39	610	1719	46	1452	1629	24	678
rc208	10	1724	51	359	1724	49	1481	1724	27	818

Πίνακας Α′.4: Αποτελέσματα για τα στιγμιότυπα του Solomon για 4 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
c101	10	1000	27	851	1010	23	509	960	15	576
c102	10	1090	24	1211	1110	19	802	1080	17	699
c103	10	1150	28	816	1160	25	841	1140	18	685
c104	10	1220	19	2037	1200	22	1108	1230	17	1148
c105	10	1030	26	1560	1060	23	660	1020	20	487
c106	10	1040	25	932	1060	26	733	1030	22	758
c107	10	1100	23	789	1110	20	937	1080	19	779
c108	10	1100	23	1309	1100	24	808	1080	24	875
c109	10	1180	22	1323	1160	20	1207	1160	19	997
c201	10	1810	40	318	1810	45	1408	1810	32	788
c202	10	1810	57	302	1810	43	1436	1810	32	740
c203	10	1810	51	303	1810	40	1455	1810	29	802
c204	10	1810	56	293	1810	40	1565	1810	30	823
c205	10	1810	51	297	1810	40	1446	1810	33	804
c206	10	1810	55	257	1810	37	1450	1810	32	788
c207	10	1810	53	290	1810	39	1439	1810	33	810
c208	10	1810	48	286	1810	40	1453	1810	32	768
r101	10	601	19	869	598	18	248	576	13	268
r102	10	807	22	1008	806	20	659	794	15	561
r103	10	878	22	862	888	22	894	858	14	603
r104	10	941	23	1282	951	16	921	925	17	800
r105	10	735	24	679	753	22	674	712	16	420
r106	10	870	20	795	881	20	907	845	17	675
r107	10	927	21	1018	912	19	1054	907	16	785
r108	10	982	17	1071	967	18	980	959	17	809
r109	10	866	18	1148	874	22	920	854	16	843
r110	10	870	22	1048	879	20	815	867	15	736
r111	10	935	20	2453	924	20	1059	905	16	717
r112	10	939	18	2248	949	20	1301	933	17	1033
r201	10	1458	75	403	1458	70	1527	1339	36	966
r202	10	1458	65	340	1458	58	1646	1392	37	775
r203	10	1458	60	284	1458	56	1636	1456	32	879
r204	10	1458	57	198	1458	40	1714	1458	30	831
r205	10	1458	69	259	1458	63	1591	1451	32	848
r206	10	1458	62	219	1458	60	1547	1458	36	1106
r207	10	1458	67	186	1458	54	1591	1458	34	802
r208	10	1458	49	133	1458	31	1551	1458	28	876
r209	10	1458	59	214	1458	49	1691	1458	34	870
r210	10	1458	66	270	1458	62	1626	1458	32	751
r211	10	1458	56	186	1458	51	1395	1458	34	791
rc101	10	794	16	1098	779	13	348	776	11	439
rc102	10	881	19	1380	878	13	615	855	14	421
rc103	10	947	15	1060	965	13	841	931	13	553
rc104	10	1019	14	1605	1024	16	797	1014	13	652
rc105	10	841	16	734	826	15	494	814	14	416
rc106	10	874	16	879	866	14	703	852	13	538
rc107	10	951	15	1147	949	15	590	956	12	624
rc108	10	998	14	2110	988	18	811	975	16	877
rc201	10	1724	63	427	1724	65	1562	1587	34	839
rc202	10	1724	67	342	1724	57	1489	1640	34	1019
rc203	10	1724	63	287	1724	49	1601	1719	33	1202
rc204	10	1724	49	201	1724	40	1524	1724	33	829
rc205	10	1724	59	364	1724	56	1561	1593	31	720
rc206	10	1724	59	295	1724	65	1526	1724	31	838
rc207	10	1724	57	300	1724	61	1553	1718	30	1026
rc208	10	1724	55	297	1724	53	1627	1724	31	805

Πίνακας Α′.5: Αποτελέσματα για τα στιγμιότυπα των Cordeau et al. για 1 διαδρομή

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
pr01	4	304	8	109	304	8	77	236	4	44
pr02	9	385	5	371	389	8	211	374	4	112
pr03	14	384	9	343	393	11	298	349	6	161
pr04	19	447	18	879	464	9	473	425	7	236
pr05	24	576	14	1221	552	12	790	446	7	353
pr06	28	538	13	1223	554	16	879	472	9	516
pr07	7	291	6	120	291	5	132	291	5	79
pr08	14	463	7	615	446	5	314	397	5	164
pr09	21	461	10	685	468	8	497	442	5	303
pr10	28	539	13	1235	528	14	933	492	7	494
pr11	4	330	5	136	340	8	97	321	3	86
pr12	9	431	5	355	434	5	293	408	3	141
pr13	14	450	7	476	447	8	426	421	6	237
pr14	19	482	9	749	505	6	687	465	8	312
pr15	24	638	17	1531	636	18	1171	594	9	409
pr16	28	559	10	4195	577	11	1290	525	6	591
pr17	7	346	9	247	349	7	194	331	4	93
pr18	14	479	10	821	523	12	401	408	6	188
pr19	21	499	9	1131	510	8	738	487	8	424
pr20	28	570	16	1844	595	14	1264	522	12	569

Πίνακας Α′.6: Αποτελέσματα για τα στιγμιότυπα των Cordeau et al. για 2 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
pr01	4	471	11	128	493	13	159	437	6	84
pr02	9	660	12	696	669	18	416	631	9	214
pr03	14	714	17	954	713	21	847	635	11	334
pr04	19	863	20	2197	832	20	1203	797	16	593
pr05	24	1011	32	3459	1047	23	2486	918	17	984
pr06	28	997	24	4219	964	24	2086	869	17	1149
pr07	7	552	7	422	555	10	301	552	8	137
pr08	14	796	19	1165	783	17	737	722	10	360
pr09	21	867	22	2592	816	26	1208	749	13	737
pr10	28	1004	26	4475	1058	26	2481	932	18	1380
pr11	4	542	10	155	521	10	218	528	6	135
pr12	9	727	10	641	727	16	616	686	6	367
pr13	14	757	14	1448	799	17	960	750	8	499
pr14	19	925	22	2432	943	17	1695	867	17	897
pr15	24	1126	29	6147	1101	28	3109	981	14	1453
pr16	28	1110	26	5159	1076	24	3199	929	14	1509
pr17	7	624	11	935	620	10	398	594	7	254
pr18	14	877	13	1152	892	21	1259	790	11	490
pr19	21	955	21	3242	899	18	2083	855	12	1020
pr20	28	1056	24	4568	1110	24	3003	950	15	1586

Πίνακας Α′.7: Αποτελέσματα για τα στιγμιότυπα των Cordeau et al. για 3 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
pr01	4	598	20	84	582	12	178	582	9	180
pr02	9	899	21	799	882	19	705	821	12	407
pr03	14	946	29	1978	935	26	1355	887	17	978
pr04	19	1195	31	6425	1201	32	2121	1146	24	1607
pr05	24	1356	39	6107	1382	42	4404	1308	21	2460
pr06	28	1376	43	12652	1353	34	3979	1220	28	2398
pr07	7	713	13	898	721	16	465	693	10	265
pr08	14	1082	27	2052	1075	28	1479	960	20	674
pr09	21	1144	29	6240	1203	27	2843	1067	19	1334
pr10	28	1473	38	11768	1446	41	4984	1312	31	2708
pr11	4	632	13	76	635	11	217	617	9	132
pr12	9	902	19	756	927	16	1282	899	10	502
pr13	14	1046	20	2150	1045	22	1877	984	15	958
pr14	19	1197	29	4614	1240	32	3304	1210	22	1951
pr15	24	1488	36	6891	1477	34	5465	1426	21	3119
pr16	28	1478	36	8589	1501	29	5878	1355	23	3808
pr17	7	808	20	344	793	13	554	781	12	338
pr18	14	1165	22	2078	1133	24	1783	1050	17	964
pr19	21	1238	27	5001	1331	29	4142	1241	21	2409
pr20	28	1514	31	14826	1495	33	6340	1402	26	4079

Πίνακας Α′.8: Αποτελέσματα για τα στιγμιότυπα των Cordeau et al. για 4 διαδρομές

		ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
pr01	4	644	17	72	654	17	297	649	10	203
pr02	9	1014	23	807	1020	20	1303	947	16	826
pr03	14	1162	27	3153	1181	26	2727	1087	21	1346
pr04	19	1452	40	6826	1453	37	3245	1343	27	2713
pr05	24	1665	56	19036	1670	49	5821	1588	39	3559
pr06	28	1696	50	10613	1711	48	6374	1544	32	5576
pr07	7	840	19	436	834	20	576	800	12	511
pr08	14	1267	37	2392	1272	34	2393	1199	23	1064
pr09	21	1460	43	6910	1507	46	4527	1421	30	2812
pr10	28	1782	56	11975	1785	50	9202	1614	39	5001
pr11	4	654	17	57	654	17	234	654	13	209
pr12	9	1041	23	761	1054	21	1723	1056	16	812
pr13	14	1263	33	3564	1255	34	2307	1157	21	1269
pr14	19	1528	34	8541	1583	37	5981	1461	25	2948
pr15	24	1818	52	9163	1805	41	7763	1658	24	4599
pr16	28	1889	40	16763	1870	41	12066	1740	26	6058
pr17	7	889	22	291	881	18	619	873	13	404
pr18	14	1352	28	3032	1384	29	2590	1318	23	1663
pr19	21	1560	39	7442	1636	34	6206	1487	27	3980
pr20	28	1846	47	13714	1867	46	8963	1784	35	6845

Πίνακας Α′.9: Αποτελέσματα για τα t1* στιγμιότυπα

			ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Διαδρ	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
t101	10	1	387	9	267	387	8	241	375	7	168
t102	10	2	772	10	593	763	11	905	766	11	455
t103	16	2	786	10	1177	803	13	1143	797	12	806
t104	12	2	737	9	824	748	10	744	739	9	497
t105	15	1	433	8	329	428	7	384	433	8	378
t106	18	3	1167	16	2205	1179	16	3039	1164	16	1788
t107	13	2	787	15	1377	759	12	883	773	14	769
t108	15	2	711	16	1428	708	15	1028	712	16	868
t109	10	3	1114	16	1516	1097	14	1305	1069	16	815
t110	16	2	807	14	1494	817	15	1132	809	15	1260
t111	19	2	821	15	1845	812	14	1257	815	15	1150
t112	14	2	800	14	740	793	13	961	821	14	867
t113	15	3	1091	19	2766	1065	17	1672	1058	17	1257
t114	12	1	467	7	334	476	7	409	456	8	277
t115	10	3	1059	12	1207	1062	14	1425	1035	13	688
t116	18	2	840	14	889	841	15	1260	839	11	1020
t117	19	1	452	9	725	446	6	638	462	8	511
t118	15	3	1140	23	1615	1133	16	1925	1140	21	1726
t119	18	3	1163	25	1732	1142	21	2416	1159	20	2214
t120	17	2	1023	15	2902	1014	16	1951	1002	14	1038
t121	16	1	424	8	302	445	6	481	428	4	331
t122	14	1	468	4	751	469	4	449	470	2	242
t123	15	1	404	5	247	410	7	339	409	8	293
t124	12	1	435	5	180	467	7	337	471	4	222
t125	18	3	1176	19	2298	1169	12	2180	1179	14	1556
t126	12	1	413	5	294	414	6	319	408	4	208
t127	11	3	1025	15	1595	1023	13	1152	1012	14	846
t128	14	3	1111	22	1994	1105	22	2027	1062	23	1273
t129	13	1	432	7	263	442	6	329	441	7	263
t130	18	2	812	18	956	804	15	1178	764	12	1023
t131	16	1	400	7	322	407	9	397	365	7	263
t132	19	1	420	10	497	416	7	509	418	7	458
t133	13	2	798	12	1395	804	17	854	804	14	660
t134	18	3	1212	24	3719	1227	24	2220	1233	21	2323
t135	16	2	823	15	2377	780	15	1083	801	15	922
t136	10	2	756	8	456	762	11	568	746	10	414
t137	17	3	1119	18	2254	1089	20	1758	1069	16	1430
t138	17	3	1222	19	2378	1203	18	2080	1206	18	2108
t139	15	3	1115	20	2257	1154	20	2149	1138	17	1500
t140	10	3	993	18	770	1035	13	1217	1004	13	660
t141	10	2	724	13	664	746	15	682	730	14	426
t142	18	3	1185	24	2516	1178	18	3289	1166	22	1907
t143	15	1	413	8	254	416	8	444	416	9	379
t144	17	2	763	14	1222	733	15	971	729	13	701
t145	10	1	357	9	196	370	6	289	364	6	151
t146	13	2	767	13	960	773	11	862	768	14	586
t147	13	3	1078	15	2124	1103	14	1306	1085	20	1288
t148	14	1	468	5	256	475	5	423	475	5	250
t149	13	3	1072	21	2070	1065	16	1230	1084	15	1214
t150	16	1	487	6	477	485	6	528	487	6	399

Πίνακας Α′.10: Αποτελέσματα για τα t2* στιγμιότυπα

			ILS			**CSCRatio**			**CSCRoutes**		
Όνομα	Clusters	Διαδρ	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)	Κέρδος	Μετακ	CPU(ms)
t201	12	1	183	3	76	187	3	107	177	2	102
t202	14	1	193	2	102	193	2	111	193	2	111
t203	10	1	174	3	67	179	4	120	178	2	95
t204	14	1	171	3	153	171	5	126	171	3	137
t205	13	3	447	10	1070	434	10	267	445	10	345
t206	17	1	196	3	212	197	7	165	196	6	201
t207	14	1	174	2	89	174	2	101	201	3	111
t208	19	1	162	3	90	176	3	138	176	3	157
t209	17	3	455	13	1010	464	13	552	446	9	480
t210	20	3	481	11	1086	482	9	527	497	9	666
t211	12	3	472	7	703	473	8	374	474	7	462
t212	10	3	461	4	394	466	5	260	460	5	234
t213	17	3	498	11	825	500	14	644	490	9	681
t214	14	2	310	6	380	322	8	283	317	6	249
t215	13	3	424	10	429	425	11	358	424	12	381
t216	12	3	463	11	600	462	10	357	468	11	328
t217	11	3	463	8	821	463	7	242	462	7	342
t218	10	1	155	1	50	155	1	69	155	2	70
t219	16	3	473	11	761	482	15	638	483	12	692
t220	13	2	334	5	592	347	4	237	329	5	185
t221	18	2	280	10	315	287	9	285	280	6	323
t222	19	2	396	10	520	396	12	456	389	11	441
t223	15	1	183	4	65	216	8	158	229	6	162
t224	13	3	402	5	539	406	6	301	405	5	282
t225	18	3	543	14	1623	551	13	648	549	12	845
t226	19	3	569	14	1151	563	14	741	578	13	819
t227	13	1	159	3	73	159	3	100	159	3	115
t228	16	3	537	11	812	531	11	697	530	11	626
t229	19	1	178	4	123	178	4	157	174	3	176
t230	11	2	288	7	176	286	11	147	285	8	193
t231	11	3	498	8	426	501	8	507	489	8	343
t232	17	3	522	15	1309	535	11	642	525	13	655
t233	15	1	180	5	112	209	6	145	213	7	163
t234	15	3	488	8	888	501	8	526	503	10	434
t235	14	3	484	12	468	496	13	366	482	11	430
t236	15	1	175	3	84	174	3	121	175	3	132
t237	11	3	473	10	954	477	9	489	479	10	491
t238	12	3	526	8	731	522	8	485	533	7	439
t239	19	3	508	15	1703	509	12	771	508	13	914
t240	12	2	297	6	183	308	8	190	322	8	232
t241	14	1	170	3	48	172	4	98	172	3	113
t242	10	1	180	2	89	180	2	91	180	2	83
t243	15	1	170	2	77	195	5	136	201	4	149
t244	10	2	331	9	176	331	8	218	332	7	183
t245	14	2	291	5	168	299	5	136	285	7	182
t246	15	3	455	11	548	453	8	459	452	9	436
t247	13	3	445	10	582	454	9	370	444	9	422
t248	17	3	445	13	941	467	13	817	465	14	629
t249	11	3	431	10	474	438	12	226	425	6	231
t250	20	1	200	7	276	201	7	174	192	5	202